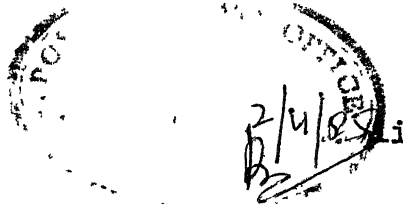


# **GOAL PROGRAMMING ALGORITHMS FOR NETWORK FLOW PROBLEMS**

**A Thesis Submitted  
In Partial Fulfilment of the Requirements  
for the Degree of  
MASTER OF TECHNOLOGY**

**by  
A. SRINIVAS**

**to the  
INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME  
INDIAN INSTITUTE OF TECHNOLOGY KANPUR  
APRIL, 1985**



CERTIFICATE

This is to certify that the present work on "Goal Programming Algorithms for Network Flow Problems," by A. Srinivas has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

(R. K. Ahuja)  
Lecturer  
Industrial and Management Engg. Program  
Indian Institute of Technology  
Kanpur 208 016, INDIA

April, 1985



12 JUN 1985

LIT KANNUK

CENTRAL

87470

IMEP-1985-M-SRI-GOA

## ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Dr. R.K. Ahuja for his valuable suggestions and excellent guidance throughout the span of this work.

I am also thankful to my friend Mr. V.V.S.N. Murthy for his help during the course of this work.

Lastly, I thank Mr. J.K. Misra for his excellent typing of this manuscript, and Mr. Buddhi Ram Kandiyal for his neat cyclostyling work.

A. Srinivas

## CONTENTS

<u>Chapter</u>		<u>Page</u>
I.	INTRODUCTION	
	1.1 Introduction	1
	1.2 Outline of the Thesis	3
II.	PRELIMINARIES	
	2.1 Introduction	7
	2.2 Graph Theory Notations	7
	2.3 Literature Review	9
	2.4 Bicriteria Linear Programming	11
	2.5 An Overview of Goal Programming	20
III.	NETWORK GOAL PROGRAMMING ALGORITHMS	
	3.1 Introduction	27
	3.2 Notations	29
	3.3 Tracing the Trade-off Curves	29
	3.4 Mathematical Formulation of WGNF Problem	36
	3.5 Development of the Algorithm	37
	3.6 Statement of Algorithm	53
	3.7 Numerical Example	58
	3.8 Mathematical Formulation of IGNF Problem	59
	3.9 Algorithm for the IGNF Problem	64
	3.10 Numerical Example	67
	3.11 Computational Results	67
	REFERENCES	77
	APPENDIX	

## ABSTRACT

In this dissertation, we consider the weighted goal programming and interval goal programming problems in the network context. Special structure embedded in their problems is used to develop computationally efficient algorithms. In weighted goal network flow [WGNF] problem, the decision maker specifies aspiration levels for each of the objectives and weighting factors for each of the deviations from the aspiration levels. On the other hand, in interval goal network flow [IGNF] problem, the decision maker provides a range of aspiration levels for each objective and weighting factors for each of the deviations from the specified range of aspiration levels. Both the algorithms developed trace a path in the feasible region and obtain the solutions which minimize the sum of weighted deviations. The WGNF and the IGNF algorithms are based on the parametric approach and fully exploit the special structure of the minimum cost flow problem in order to perform all the computations on the network itself. Computer programs were written for both the algorithms, and tested on randomly generated network problems. Results of their investigations are presented.

## CHAPTER I

### INTRODUCTION

#### 1.1 Introduction:

Operations Research is a scientific approach to solve complex problems arising in the management of large systems of men, machines, materials and money. Operations Research is a decision science which helps management to make better decisions. Network model is a branch of study in Operations Research. Due to the wide applicability of network models in real world, it is considered as an important branch of study in Operations Research. In the present day, we find that complex intriguing problems arising in production-distribution systems, military logistics systems, urban traffic systems, railway systems, communication systems, pipeline network systems, facilities location systems, file merge systems, electrical networks etc. can be tackled by constructing an appropriate network model. Furthermore, network geometry (or relationships) can be easily displayed in two-dimensional drawings, greatly simplifying the communication problem between the analyst and the client for whom the model is designed. In the recent years, network flow problems have received special attention due to significant

advances in implementation technology and solution techniques, thereby increasing the applicability of the network models substantially.

In this dissertation, we consider the bicriteria minimum cost flow problems and propose two algorithms i.e. weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms for obtaining optimum solutions. These models are useful when two conflicting objectives are to be simultaneously considered, i.e. one is interested in minimizing the total cost as well as total time.

One application of the bicriteria minimum cost flow problem arises in a distribution system. Suppose  $V$  units of perishable items (i.e. fruits) are to be sent from source to sink along the road network. The two objectives to be considered simultaneously are total cost and total time. We would like to transport the items in such a way that they are not perished before they reach the sink and at the same time the total cost of transportation should be minimum.

Another application of the bicriteria minimum cost flow problem arises in communication networks. Suppose  $V$  units of messages/sec. are required to be transmitted from source to sink along a communication network. The decision maker would like to consider two objectives simultaneously. One of the objectives may be to minimize total cost of sending messages,

and the other being the minimization of total amplification cost.

Another important application of bicriteria network flow problem is in the area of flow through a pipeline network. The decision maker would like to minimize the total operating cost for sending the flow as well as minimize the total time to transmit flow from source to sink.

In this work, we develop specializations of goal programming i.e. weighted goal and interval goal network flow algorithms based on parametric approach. The special structure of minimum cost flow problem is exploited to perform all the computations on the network itself. Efficient tree data structures are used to further enhance the efficiency of these computations. Computational investigations with the above approaches are found to be very encouraging and presented in sufficient detail.

## 1.2 Outline of the Thesis:

In this section, we give a brief outline of this thesis.

In Chapter II, we present some of the preliminary knowledge required to understand the work done in the subsequent chapters. We first review the literature related to the bicriteria minimum cost flow problem. The graph theory notations adopted in this work are also described. Finally, an

overview of the bicriteria network flow problem and the goal programming are briefly outlined.

Chapter III embodies the main work done on WGNF problem and IGNF problem. They can be formulated as shown below:

Weighted Goal Network Flow Problem:

$$\text{Min. } Z = w_1\alpha_1 + w_2\alpha_2 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{Otherwise} \end{cases}$$

$$0 < x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1$$

Interval Goal Network Flow Problem:

$$\text{Min. } Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i) \in I(1)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i = 1 \\ V, & \text{if } i = n, \forall i \in N \\ 0, & \text{Otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1$$



$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_3 - \alpha_3 = C_2$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_4 - \alpha_4 = D_2$$

where,

$x_{ij}$  is the amount of flow on arc  $(i,j)$ .

$c_{ij}$  is the cost of unit flow on arc  $(i,j)$ .

$d_{ij}$  is the budget required for unit flow on arc  $(i,j)$

$b_{ij}$  is the capacity of arc  $(i,j)$ .

$V$  is the net flow of commodity from source to sink  $n$ .

$C_i$  is the aspiration level for cost objective function.

$D_i$  is the aspiration level for budget objective function.

$\alpha_i$  is the positive deviation.

$\beta_i$  is the negative deviation.

We suggest two exact algorithms to solve bicriteria network flow problem. They are weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms. Both these algorithms are based on the parametric algorithm for constrained minimum cost flow problem [1].

The WGNF algorithm searches for an optimum solution which satisfies the aspiration levels specified for each objective by the decision maker. Whereas IGNF algorithm searches for an optima that lies in the specified range of aspiration levels for each of the objectives.

The computational investigations of both the algorithms are given at the end of Chapter III. Computer programs are written for both the algorithms and the computational performances are ascertained by solving different sized problems. Efficient data structures are implemented to represent the basis tree which requires comparatively less storage. Data structures based on augmented threaded index method are used and results of the computational investigations are presented.

## CHAPTER II

### PRELIMINARIES

#### 2.1 Introduction:

In this chapter, we review some of the relevant concepts of the bicriteria minimum cost flow problem and goal programming. The basic reason behind this is to prepare a sufficient background which will help in understanding the algorithms developed in the subsequent chapters.

This chapter is divided into five sections, The graph theory notations adopted in this work are presented in Sec. 2. The literature related to the bicriteria minimum cost flow problem is surveyed in Sec. 3. A brief review of various methods of solving bicriteria minimum cost flow problem and a parametric algorithm for solving constrained minimum cost flow problem [1] are presented in Sec. 4. In Sec. 5 a brief description about goal programming and its importance to management in decision making to real world situations, and various goal programming techniques are discussed.

#### 2.2 Graph Theory Notations:

Some notations and well-known concepts of graph theory that are being used throughout the thesis are mentioned below.

A directed graph  $G = (N, A)$ , consists of a finite set  $N$  of elements, called nodes, and a set  $A$  of ordered pairs of nodes called arcs. A directed network is a directed graph in which numerical values are attached to the nodes and arcs of the graph. Let  $n = |N|$  and  $m = |A|$ . The two specified nodes 1 and  $n$  are called the source and the sink respectively.

An arc  $(i, j)$  has two end points,  $i$  and  $j$ , and it is said to be incident from node  $i$  and incident to node  $j$ . Let  $I(i)$  and  $O(i)$  denote, respectively, the sets of arcs incident to and incident from node  $i$ . The degree of a node  $i$  is the number of arcs incident to or incident from that node.

A path in  $G = (N, A)$  is a sequence  $i_1, i_2, \dots, i_r$  of distinct nodes of  $N$  such that either  $(i_k, i_{k+1}) \in A$  or  $(i_{k+1}, i_k) \in A$  for each  $k = 1, \dots, r-1$ . A directed path is defined similarly, except that  $(i_k, i_{k+1}) \in A$  for each  $k = 1, \dots, r-1$ . A cycle is a path together with an arc  $(i_r, i_1)$  or  $(i_1, i_r)$ . A directed cycle is a directed path together with the arc  $(i_r, i_1)$ .

A graph  $G = (N', A')$  is a subgraph of  $G = (N, A)$  if  $N' \subseteq N$  and  $A' \subseteq A$ . A graph  $G = (N', A')$  is a spanning subgraph of  $G = (N, A)$  if  $N' = N$  and  $A' \subseteq A$ .

Two nodes  $i$  and  $j$  are said to be connected if there is at least one path between them. A graph is said to be

connected if all pairs of nodes are connected, otherwise it is called disconnected. A set  $Q \subseteq A$  such that the graph  $G = (N, A-Q)$  is disconnected and no subset of  $Q$  has this property, is called a cocycle of  $G$ . A cocycle is a cutset if it disconnects source and sink.

A graph is acyclic if it does not contain any cycle. A tree is a connected acyclic graph. A subtree of a tree  $T$  is a subgraph of  $T$  as well as a tree. A tree  $T$  is said to be a spanning tree of  $G$  if  $T$  is a spanning subgraph of  $G$ . Arcs belonging to a spanning tree  $T$  are called tree-arcs and arcs not belonging to  $T$  are called nontree-arcs. A spanning tree of  $G = (N, A)$  has exactly  $(n-1)$  tree-arcs.

### 2.3 Literature Review:

One of the methods of solving the bicriteria minimum cost flow problem is to consider one of the objectives as a constraint and solve the resultant constrained minimum cost flow [CMCF] problem. We will first present a brief review of the constrained flow problems.

Hultz and Klingman [9] have suggested a partitioning method in conjunction with the simplex method for solving constrained generalized network flow problems. Takashi Kobayashi [12] proposed a primal-dual method for solving CMCF problem. He has associated two dimensional distance for each arc. The first element is related to the cost and the second

one to the coefficient of the additional constraint. This method is suitable in cases when degeneracies often occur.

A parametric algorithm for solving CMCF problem is developed by Ahuja, Batra and Gupta [1]. The algorithm uses the concepts from parametric linear programming and fully exploits the topological structure embedded in the problem. The algorithm can also be used to generate all the efficient points of the bicriteria minimum cost flow problem.

Chen and Saigal [4] have suggested a primal algorithm for solving a capacitated network flow problems with additional linear constraints.

Klingman and Mote [10] have reviewed the fundamental theoretical results for the general multi-criteria linear programming and the relevant exploitable characteristics of the network basis. Two algorithms are then developed for solving the multicriteria network flow problems efficiently. One approach determines the set of all non-dominated solutions to the problem. The other approach is a network variant of the surrogate criterion linear programming approach. Ignizio [7] has suggested a straight forward weighted integer goal programming for generalized networks models for integer programming problems. His method is simplex and robust.

Practically no study was made in the field of application of goal programming techniques to network flow problems. This has motivated us to propose two algorithms for WGNF and IGNF problems.

#### 2.4 Bicriteria Linear Programming:

Bicriteria linear programming deals with the optimization of two objective functions (multicriteria deals with two or more than two objective functions) simultaneously. A decision situation is generally characterised by multiple objectives. Some of these objectives may be complementary, while others may be conflicting in nature.

The bicriteria minimum cost flow problem differs from the classical minimum cost flow problem only in the expression of their objective functions. The bicriteria minimum cost flow problem is shown as below:

$$\text{Minimize} \quad \left\{ \begin{array}{l} \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{(i,j) \in A} d_{ij} x_{ij} \end{array} \right\}$$

subject to

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A$$

Generally, it is observed that these two objectives are conflicting in nature and hence there is no optimal solution in the normal sense for the above problem. The decision maker has to choose solutions, possibly not the best for both the criteria. A special set of solutions, the non-dominated or efficient solutions can be defined to overcome this problem.

An efficient solution is one in which one objective cannot be reduced without a simultaneous increment in the other objective. That is,  $X^*$  is an efficient solution to the bicriterion minimum cost flow problem if there does not exist any  $X^0 \in S$ , the set of all possible solutions such that

$$\begin{aligned} Z_1(X^0) &\leq Z_1(X^*) \quad \text{and} \\ Z_2(X^0) &\leq Z_2(X^*) \end{aligned}$$

with atleast one strict inequality.

A number of techniques as mentioned below are available for generating efficient or non-inferior solutions for the bicriteria problem formulated above.

- (i) The weighting method
- (ii) The constraint method
- (iii) The non-inferior set estimation method, and
- (iv) The multi-objective simplex algorithm.

These techniques are discussed in detail by Ambrose Goicoechea [3].



There is one disadvantage with non-dominated solutions technique. It becomes difficult for the decision maker to make his final choice from a set of non-dominated solutions. Another technique known as goal programming, allows the decision maker to specify a target for each of the objective functions. It obtains a preferred solution which is defined as the one that minimizes the sum of the deviations from the prescribed set of target values. A brief description of the goal programming is presented in Sec. 2.5.

#### An Overview of Constrained Minimum Cost Flow Problems

We now briefly outline the parametric algorithm for the constrained minimum cost flow [CMCF] problem which is the basis for the algorithms developed in this thesis.

The mathematical formulation of the CMCF problem is

$$\text{Min. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (2.1)$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i,j) \in A \quad (2.3)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \leq D \quad (2.4)$$

This algorithm is developed by Ahuja, Batra and Gupta [1]. The algorithm utilizes the concepts of parametric linear programming

and performs all the computations over the network. It first obtains the optimum basis corresponding to  $C_{\min}$ , where  $C_{\min}$  is the minimum value of  $C$  for which a feasible flow exists and then moves from one optimum basis to the next as  $D$  is decreased. The algorithm thus yields the minimum cost as a function of the budget, which is a piecewise linear convex function.

### Optimality Conditions:

It is well known that the optimum basis of MCF problem is a spanning tree [5]. The addition of the budget constraint introduces one more arc in the optimum basis. It follows from bounded variable linear programming that the necessary and sufficient conditions for a feasible basis structure to be an optimum basis structure are that there exist dual variables  $\pi_j$ 's and  $\mu \geq 0$ , satisfying the following conditions:

$$(i) \quad \pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in B \quad (2.5)$$

$$(ii) \quad \pi_j - \pi_i \leq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in L \quad (2.6)$$

$$(iii) \quad \pi_j - \pi_i \geq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \in U \quad (2.7)$$

$$(iv) \quad \mu (D - \sum_{(i,j) \in A} d_{ij} x_{ij}) = 0 \quad (2.8)$$

where  $B$ ,  $L$ , and  $U$  represent the sets of arcs corresponding to the basic variables, the nonbasic variables at their lower bounds, and the nonbasic variables at their upper bounds,

respectively. The set  $B$  is called a basis and the triple  $(B, L, U)$  is called a basis structure.

Let  $(T \cup \{(p, q)\}, L, U)$  be an optimum basis structure at  $D = D'$ . Further let  $\pi_j^c$  and  $\pi_j^d$  be the numbers satisfying

$$\pi_1^c = 0 \quad \text{and} \quad \pi_j^c - \pi_i^c = c_{ij}, \quad \forall (i, j) \in T \quad (2.9)$$

$$\pi_1^d = 0 \quad \text{and} \quad \pi_j^d - \pi_i^d = d_{ij}, \quad \forall (i, j) \in T \quad (2.10)$$

Define the numbers  $\bar{c}_{ij}$  and  $\bar{d}_{ij}$  for all arcs  $(i, j) \in L \cup U$  as follows:

$$\bar{c}_{ij} = \begin{cases} \pi_i^c - \pi_j^c + c_{ij}, & \text{if } (i, j) \in L \\ \pi_j^c - \pi_i^c - c_{ij}, & \text{if } (i, j) \in U \end{cases} \quad (2.11)$$

$$\bar{d}_{ij} = \begin{cases} \pi_i^d - \pi_j^d + d_{ij}, & \text{if } (i, j) \in L \\ \pi_j^d - \pi_i^d - d_{ij}, & \text{if } (i, j) \in U \end{cases} \quad (2.12)$$

Clearly,  $\bar{c}_{ij} = \bar{d}_{ij} = 0$ , for all  $(i, j) \in T$ . If  $\mu$  is a real number and  $\pi_j$  be the numbers satisfying,

$$\pi_1 = 0 \quad \text{and} \quad \pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall (i, j) \in T$$

then,

$$\pi_j = \pi_j^c + \mu \pi_j^d, \quad \forall j \in N \quad (2.13)$$

substituting (2.13) in the conditions (2.5) - (2.8) and then using (2.11) and (2.12), we obtain the following equivalent conditions:

$$(i) \quad \bar{c}_{pq} + \mu \bar{d}_{pq} = 0 \quad (2.14)$$

$$(ii) \quad \bar{c}_{ij} + \mu \bar{d}_{ij} \geq 0 \quad \forall (i,j) \in L \cup U \quad (2.15)$$

$$(iii) \quad \mu (D' - \sum_{(i,j) \in A} d_{ij} x_{ij}) = 0 \quad (2.16)$$

The conditions (2.14) - (2.16) are subsequently referred to as the optimality conditions. The  $\bar{c}_{ij}$ ,  $\bar{d}_{ij}$  and  $\mu$  are referred to as the cost price, budget price and price ratio, respectively, associated with the basis structure  $(T \cup \{(p,q)\}, L, U)$ .

Depending upon the values of  $\bar{c}_{ij}$  and  $\bar{d}_{ij}$ , arcs in  $L \cup U$  can be classified as follows:

- (i) active arcs:  $\{(i,j) \in L \cup U: \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} < 0\}$ .
- (ii) critical arcs:  $\{(i,j) \in L \cup U: \bar{c}_{ij} \leq 0 \text{ and } \bar{d}_{ij} < 0\}$ .
- (iii) passive arcs:  $\{(i,j) \in L \cup U: \bar{d}_{ij} \geq 0\}$ .

The piecewise linear convex curve between cost and budget is obtained as follows.

Firstly an arc  $(p, q)$  belonging to the critical set is selected and added to basis. This results in the formation of exactly one cycle. Next the flow is augmented in this cycle. One of the arcs in the cycles reaches one of its bounds and leaves the basis respectively. The dual variables as well as  $(B, L, U)$  are updated and the step is repeated until the critical set is empty. We now describe how the algorithm moves from one basis to the next basis satisfying

the optimality conditions while decreasing the total budget  $D$ .

Addition of an arc  $(i,j) \in L \cup U$  to the basic tree creates exactly one cycle  $W_{ij}$  consisting of the basic arcs. We define the orientation of the cycle  $W_{ij}$  along  $(i,j)$  if  $(i,j) \in L$  and opposite to  $(i,j)$  if  $(i,j) \in U$ . Let  $\bar{W}_{ij}$  and  $\underline{W}_{ij}$  be the sets of arcs in the cycle  $W_{ij}$  along and opposite to its orientation, respectively. Then using (2.9) and (2.10) it can be easily shown that

$$\bar{c}_{ij} = \sum_{(i,j) \in \bar{W}_{ij}} c_{ij} - \sum_{(i,j) \in \underline{W}_{ij}} c_{ij} \quad (2.17)$$

$$\bar{d}_{ij} = \sum_{(i,j) \in \bar{W}_{ij}} d_{ij} - \sum_{(i,j) \in \underline{W}_{ij}} d_{ij} \quad (2.18)$$

Thus,  $\bar{c}_{ij}$  (or  $\bar{d}_{ij}$ ) denotes the increase in the cost (or budget) if unit amount of additional flow is circulated in the cycle  $W_{ij}$  along its orientation. The above given classification of non-tree arcs can be given the physical interpretation. Active arcs are those arcs which can lead to increase in cost if budget is decreased. Critical arcs are those which do not increase cost if budget is reduced. Passive arcs do not lead to decrease in budget even if more cost occurs.

#### Characteristic Interval:

Let at  $D = \underline{D}$ , the optimum basis structure of the CMCF problem is  $(T \cup \{(p,q)\}, L, U)$ . Let  $\underline{x}_{ij}$  be the flow on arc  $(i,j) \in A$  and  $\underline{z}$  be the cost of this flow. Further, let

$c_{ij}$ ,  $\bar{d}_{ij}$  and  $\bar{c}_{ij}$  be the cost prices, budget prices and price ratio associated with the given basis structure.

We now determine the interval  $(\underline{D}, \bar{D})$  for the values of  $D$  for which the given basis structure continues to remain optimum. This interval is known as the characteristic interval associated with  $(T \cup \{(p, q)\}, L, U)$ .

Since the numbers,  $\bar{c}_{ij}$ ,  $\bar{d}_{ij}$  and  $\bar{c}_{ij}$  are uniquely determined for a given basis structure, the optimality conditions (2.14) and (2.15) are not affected by decrease in the value of  $D$ . However, since  $\mu > 0$ , the flow must be changed in order to satisfy (2.16). The only way to change the flow, without changing the basis structure and satisfying the flow conservation constraints (2.2), is by circulating some additional flow in the cycle  $W_{pq}$  along its orientation. It was noted that  $\bar{d}_{pq} < 0$  is the rate at which the budget is reduced and  $\bar{c}_{pq} > 0$  is the rate at which the additional cost is incurred when unit amount of flow is circulated. Since the changed flow must also satisfy the bound restrictions of the arcs (2.3), we calculate the maximum flow  $\bar{f}$  that can be circulated without violating the bound restrictions of the arcs in  $W_{pq}$ . If we define,

$$\bar{f}_{ij} = \begin{cases} b_{ij} - x_{ij} , & \text{if } (i, j) \in W_{pq} \\ x_{ij} , & \text{if } (i, j) \in \bar{W}_{pq} \end{cases}$$

then,

$$\bar{f} = \min_{(i,j) \in W_{pq}} \{f_{ij}\}$$

Thus,

$$\bar{D} = \underline{D} + \bar{f} \bar{d}_{pq}$$

For all values of  $D$  in  $(\underline{D}, \bar{D})$ , the optimum flow is given by,

$$x_{ij} = \begin{cases} x_{ij} + \alpha' \bar{f}, & \text{if } (i,j) \in W_{pq} \\ x_{ij} - \alpha' \bar{f}, & \text{if } (i,j) \in W_{pq} \\ x_{ij}, & \text{otherwise} \end{cases}$$

and cost by

$$Z = \underline{Z} + \alpha' \bar{f} \bar{c}_{pq},$$

where,  $\alpha' = (D - \underline{D})/(\bar{D} - \underline{D})$

Let  $\bar{x}_{ij}$  denote the flow in arc  $(i,j) \in A$  at  $D = \bar{D}$ . If it is required to find the optimum flow for  $D < \bar{D}$ , then a dual simplex iteration is performed to obtain a new basis structure at  $D = \bar{D}$ .

#### The Dual Simplex Iteration:

At  $D = \bar{D}$ , flow in an arc  $(u, v) \in W_{pq}$ , for which  $\bar{f}_{uv} = \bar{f}$ , equals its lower or upper bound. If  $D$  is decreased further, the bound is violated. Thus, to obtain a new basis structure at  $D = \bar{D}$ , which may permit decrease in the value of  $D$ , the arc  $(u, v)$  is dropped from the basis and a non-basic arc

is selected to enter the basis. The arc  $(u, v)$  becomes a non-basic arc at its respective bound. In the new basis structure, the dashed values and sets represent the corresponding values and sets of the previous basis structure. By selecting an active arc and performing dual simplex iteration each time until the active set is empty, the cost is increased and the budget is reduced.

The curve starts at  $C_{\min}$  and as the value of  $D$  is decreased, the value of  $Z$  keeps increasing until a value of  $D$  is reached when  $Z$  stops increasing. The slope of the curve at any point is the value of  $-\mu$  in the optimum solution which corresponds to that point. Since the value of  $-\mu$  keeps increasing and finally becomes zero and no critical arcs are formed at any iteration (The proof is given in [1]), the curve is a piecewise linear convex function.

## 2.5 An Overview of Goal Programming:

As mentioned in the previous section, Goal programming allows the decision maker to specify a target for each objective function which provides him the preferred solution by minimizing the sum of the deviations from the prescribed set of target values.

Generalized goal programming has a number of special terms and concepts that are being used in this thesis. They are mentioned as below:



(i) Objective:

An objective is a relatively general statement (in **narrative** or quantitative terms) that reflects the desires of the decision maker. For example, one may wish to "maximize profit" or "minimize total wastage" or "wipe out poverty".

(ii) Aspiration Level:

An aspiration level is a specific value associated with the desired or acceptable level of achievement of an objective. Thus, an aspiration level is used to measure the achievement of an objective and generally serves to "anchor" the objective to reality.

(iii) Goal:

An objective in conjunction with an aspiration level is termed a goal. For example, we may wish to "achieve at least X units of profits" or "reduce the rate of inflation by Y percent."

(iv) Goal Deviation:

The difference between what one accomplishes and what one aspires to is the deviation from his goal. A deviation can represent over as well as under achievement of a goal.

(v) Goal Formulation:

We will now examine how to mathematically transform an objective into a goal within our goal programming frame work.

Consider the objective function expressed in general terms as  $f_i(X)$ . The procedure to be presented is applicable whether  $f_i(X)$  is linear or nonlinear, but only linear objectives are considered in this thesis.

We then let,

$f_i(X)$  = mathematical representation of objective  $i$   
as a function of the decision variables  
 $X = (x_1, x_2, \dots, x_n)$

$b_i$  = value of the aspiration level associated  
with objective  $i$ .

Three possible forms of goals may then result:

- (i)  $f_i(X) \leq b_i$ : that is, we wish to have a value of  $f_i(X)$  that is equal to or less than  $b_i$ .
- (ii)  $f_i(X) \geq b_i$ : that is, we wish to have a value of  $f_i(X)$  that is equal to or greater than  $b_i$ .
- (iii)  $f_i(X) = b_i$ : that is  $f_i(X)$  must exactly equal  $b_i$ .

Regardless of the form, we shall transform any of these relations into the goal programming format by adding

a negative deviation variable ( $\beta_j \leq 0$ ) and subtracting a positive deviation variable ( $\alpha_i \geq 0$ ). This statement is summarized as below:

Goal Type	Goal Programming Form	Deviation Variables to be Minimized
$f_i(X) \leq b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	$\alpha_i$
$f_i(X) \geq b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	$\beta_i$
$f_i(X) = b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	$\alpha_i + \beta_i$

We will briefly discuss the three extensions of goal programming, They are:

- (1) Weighted goal programming.
- (2) Interval goal programming.
- (3) Fuzzy goal programming.

#### Weighted Goal Programming:

In this model the decision maker assigns an aspiration level for each of the objectives and also the weighting factors for each of the deviations. It obtains an optimum solution by minimizing the sum of weighted deviations.

There is an alternative way in which the weighted model may be formed. Rather than multiplying each deviation variable by a constant weight, we may instead, raise each

deviation variable in the achievement function to some power. This results in a polynomial form for the achievement function.

Given a multi-objective model,

optimize  $Z_i \quad i = 1, \dots, s$

s.t.

$$AX \leq b$$

$$X \geq 0$$

Adding aspiration levels and deviations variables to each objective and weighting each resultant goal, we obtain,

$$\text{Minimize } a = \sum_{i=1}^s (w_i \alpha_i + r_i \beta_i)$$

subject to

$$Z_i(X) + \beta_i - \alpha_i = Z_i^0, \quad i = 1, \dots, s$$

$$AX \leq b$$

$$X, \alpha_i, \beta_i \geq 0, \quad i = 1, \dots, s$$

where,

$\alpha_i$  = Weighting factor for the positive deviation of goal  $i$ .

$\beta_i$  = Weighting factor for the negative deviation of goal  $i$ .

$Z_i^0$  = Aspiration level for objective  $i$ .

### Interval Goal Programming:

In this model the decision maker specifies a range of aspiration levels for each of the goals instead of one

aspiration level. It obtains an optimum solution by minimizing the weighted sum of deviations from the set of ranges.

The mathematical formulation of this model is,

$$\text{Minimize } a = \sum_{i=1}^s (w_{i,2} \alpha_{i,2} + r_{i,1} \beta_{i,1})$$

subject to

$$Z_{i,1} + \beta_{i,1} - \alpha_{i,1} = L_1 \quad \text{all } s$$

$$Z_{i,2} + \beta_{i,2} - \alpha_{i,2} = U_1 \quad \text{all } s$$

where,

$$Z_{i,1} = Z_{i,2} = \text{expression for objective } k.$$

$$\beta_{i,1}, \beta_{i,2} = \text{negative deviations.}$$

$$\alpha_{i,1}, \alpha_{i,2} = \text{positive deviations.}$$

$$r_{i,1} = \text{weighting factor for the negative deviation for goal } Z_{i,1}.$$

$$w_{i,2} = \text{weighting factor for the positive deviation for goal } Z_{i,2}.$$

### Fuzzy Linear Programming.

A fairly recent attempt at modeling and solving the multiple-objective problem is that known as fuzzy programming. The approach is similar, in many respects, to the weighted linear goal programming method previously discussed, differing primarily in the manner in which the importance of the goals

are considered. This method minimizes the worst under achievement of any goal.

A major advantage of fuzzy linear programming is that it may be transformed into a conventional linear programming model. The main disadvantage of this method is, the underachievement of just one goal can have a major impact on the solution, since it attempts to minimize the maximum underachievement.

## CHAPTER III

### NETWORK GOAL PROGRAMMING ALGORITHMS

#### 3.1 Introduction:

In this chapter, we shall consider two special classes of linear goal programming problems, i.e. weighted goal network flow problem and interval goal network flow problem.

We generally confront problems that require two objectives to be considered simultaneously. Often, these objectives are conflicting in nature. These types of problems may arise in road networks, communication networks and pipe line networks. The typical applications of bi-criteria network flow problems are mentioned in Section 1.1.

One of the methods of solving bi-criteria problems is to obtain the efficient or non-dominated solutions set. As the efficient solution set is considerably large, it becomes practically difficult for the decision maker to choose the solution he would prefer.

Goal programming is a technique which takes the decision maker's preference into consideration and provides him the preferred solution. He expresses his preference by specifying targets to the objectives and the weights to the deviations from the targets.

In weighted goal programming, the decision maker specifies a target for each of the objective and weights for the deviations. It provides him an optimum solution which is as close as possible to the specified targets, by minimizing the sum of weighted deviations.

In interval goal programming the decision maker specifies a range or interval for each of the objectives and weights for each of the deviations from lower and upper bounds of the interval. It obtains an optimum solution lying in the ranges specified if one exists or finds a solution such that the sum of weighted deviations are minimum.

This chapter has been divided into 11 sections. In Sec. 2 we present the notations used to represent the feasible region of the bi-criteria network flow problem. The tracing of various trade-off curves is presented in Sec. 3. The mathematical formulation and development of the WGNF algorithm are presented in Sec. 4 and 5. In Sec. 6 and Sec. 7 we present the statement of the WGNF algorithm and a numerical example. The mathematical formulation and development of IGNF algorithm are given in Sec. 8 and Sec. 9. A numerical example of IGNF problem is given in Sec. 10. Finally the computational performance of both the algorithms is reported in Sec. 11.



### 3.2 Notations:

The various regions as shown in Fig. 3.1 are defined as below:

- $$\begin{aligned}
 E &: \{(x,y) : x \geq 0 \text{ and } y \geq 0\}. \\
 S &: \{(x,y) : x = \sum_{(i,j) \in A} c_{ij} x_{ij}, \quad y = \sum_{(i,j) \in A} d_{ij} x_{ij}\}. \\
 R_1 &: \{(x,y) : 0 \leq x \leq f_1, \quad 0 \leq y \leq g_2 \text{ and } (x,y) \notin S\} \\
 R_2 &: \{(x,y) : 0 \leq x \leq f_3, \quad g_2 \leq y \leq \infty \text{ and } (x,y) \notin S\} \\
 R_3 &: \{(x,y) : f_3 \leq x \leq \infty, \quad g_4 \leq y \leq \alpha \text{ and } (x,y) \notin S\} \\
 R_4 &: \{(x,y) : f_1 \leq x \leq \infty, \quad 0 \leq y \leq g_4 \text{ and } (x,y) \notin S\} \\
 S_1 &: \{(x,y) \in S : x \leq f_1\} \\
 S_2 &: \{(x,y) \in S : x \leq f_3\} \\
 S_3 &: \{(x,y) \in S : x > f_3\} \\
 S_4 &: \{(x,y) \in S : x > f_1\} \\
 \alpha_i &: \text{positive deviation from goal } i. \\
 \beta_i &: \text{negative deviation from goal } i. \\
 w_i &: \text{weight assigned to positive deviation } \alpha_i. \\
 r_i &: \text{weight assigned to negative deviation } \beta_i. \\
 w_i, r_i &> 0.
 \end{aligned}$$

### 3.3 Tracing the Trade-Off Curves:

The optimum solutions for WGNF problem and IGNF problem lie either on  $B_1$ ,  $B_2$ ,  $B_3$  and  $B_4$  curves or in the feasible region  $S$  as shown in Fig. 3.1. Since our algorithm searches for an optimum solution by traversing along  $B_1 \cup B_2 \cup B_3 \cup B_4$

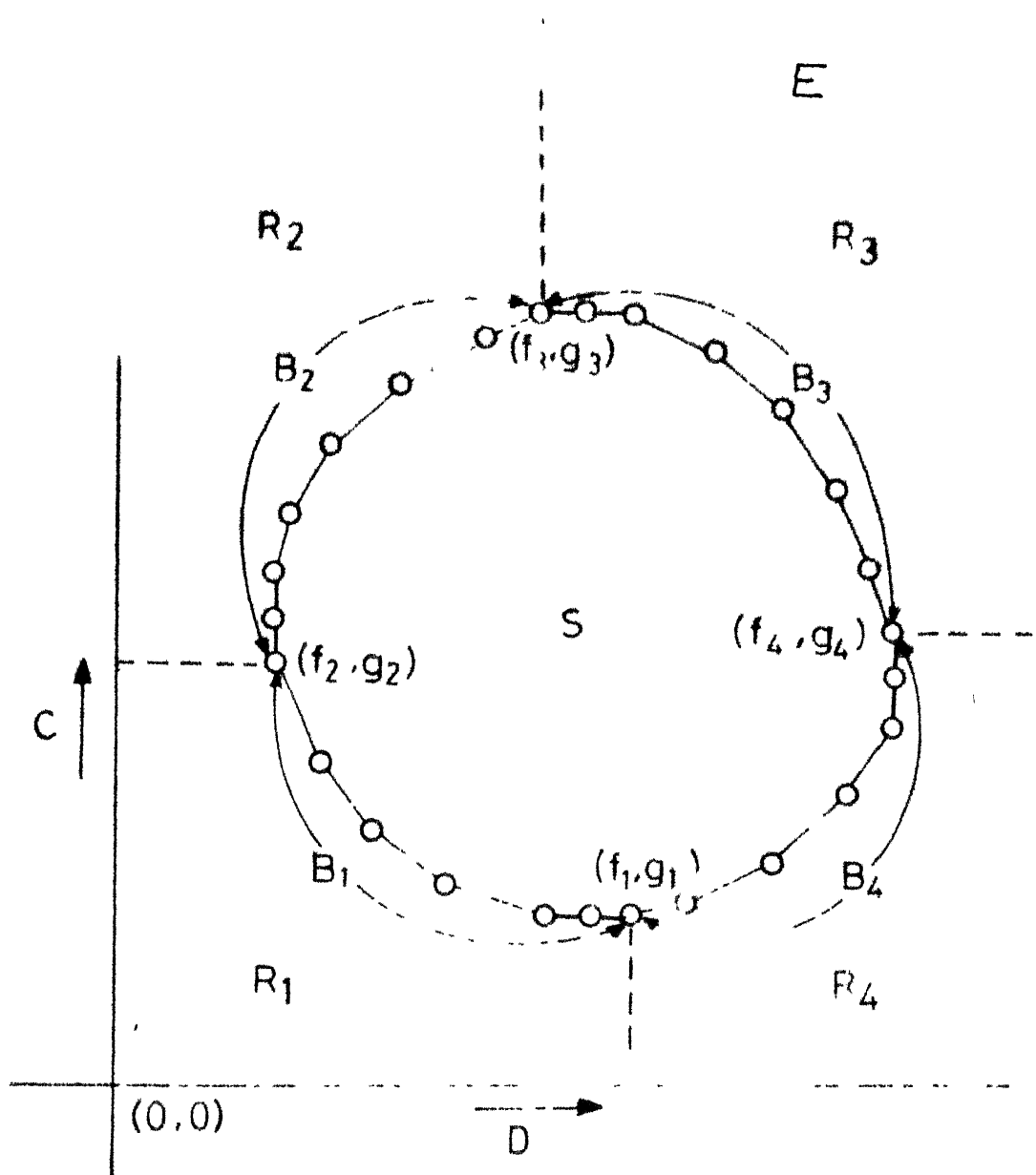


Fig. 3 1 Feasible region of bicriteria network flow problem

and through the feasible region  $S$  it is necessary to trace the  $B_1$ ,  $B_2$ ,  $B_3$  and  $B_4$  curves. Once these curves are obtained, it can be shown that they together form a closed convex feasible region  $S$ .

### Tracing $B_1$ Curve:

The detailed procedure for obtaining the  $B_1$  curve was presented in Sec. 2.4. The procedure for tracing  $B_2$ ,  $B_3$  and  $B_4$  curves is same as that for tracing  $B_1$ , except that the optimality conditions, the critical set, the active set, the passive set and the criteria for selecting the price ratio differ, as the problems considered to obtain each of these curves are different.

### Tracing $B_2$ Curve:

$B_2$  curve is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the following problem.

$$\text{Max. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

subject to

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} < D$$

The optimality conditions are:

$$(i) \quad \bar{c}_{pq} - \mu \bar{d}_{pq} = 0$$

$$(ii) \quad -\bar{c}_{ij} + \mu \bar{d}_{ij} \geq 0 \quad \forall (i,j) \in L \cup U$$

$$(iii) \quad \mu(D - \sum_{(i,j) \in A} d_{ij} x_{ij}) = 0$$

The non-basic arcs are classified as:

$$(i) \quad \text{Active arcs } S' : \{(i,j) \in L \cup U : \bar{c}_{ij} < 0 \text{ and } \bar{d}_{ij} < 0\}$$

$$(ii) \quad \text{Critical arcs } S'' : \{(i,j) \in L \cup U : \bar{c}_{ij} \geq 0 \text{ and } \bar{d}_{ij} < 0\}$$

$$(iii) \quad \text{Passive arcs } S''' : \{(i,j) \in L \cup U : \bar{d}_{ij} \geq 0\}$$

Initially,  $\sum_{(i,j) \in A} c_{ij} x_{ij}$  is maximized and let

$(C_{\max}, D)$  be the solution. A critical arc  $(p,q) \in S''$  is

selected and entered into the basis and dual-simplex iteration is performed. This is continued until  $S$  is empty.

Next, an active arc  $(p,q)$  that that  $\mu_{pq} = \min_{(i,j) \in S'} (\mu_{ij})$  is selected and entered into the basis and dual-simplex iteration is performed. Each dual-simplex iteration results in an extreme solution. This is repeated until  $S' = \{\emptyset\}$ . As  $\mu$  is gradually increasing and ultimately becomes zero along with gradual decrease in  $C$  and  $D$ , the curve  $B_2$  is a piecewise concave linear function.

Tracing  $B_3$  Curve.

$B_3$  is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the

following problem.

$$\begin{aligned} \max. \quad Z &= \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{(j,1) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \\ & 0 \leq x_{ij} \leq h_{ij} \quad \forall (i,j) \in A \\ & \sum_{(i,j) \in A} d_{ij} x_{ij} \geq D \end{aligned}$$

The optimality conditions are:

- (i)  $\bar{c}_{pq} + \mu \bar{d}_{pq} = 0$
- (ii)  $-\bar{c}_{ij} - \mu \bar{d}_{ij} \geq 0 \quad \forall (i,j) \in A$
- (iii)  $\mu \left( \sum_{(i,j) \in A} d_{ij} x_{ij} - D \right) = 0$

The non basic-arcs are classified as:

- (i) Active arcs  $S'$  :  $\{(i,j) \in L \cup U : \bar{c}_{ij} < 0 \text{ and } \bar{d}_{ij} > 0\}$
- (ii) Critical arcs  $S''$  :  $\{(i,j) \in L \cup U : \bar{c}_{ij} \geq 0 \text{ and } \bar{d}_{ij} > 0\}$
- (iii) Passive arcs  $S'''$  :  $\{(i,j) \in L \cup U : \bar{d}_{ij} \leq 0\}$

Initially,  $\sum_{(i,j) \in A} c_{ij} x_{ij}$  is maximized and let  $(C_{\max}, D)$  be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc  $(p,q)$  such that

$$\mu_{pq} = \min_{(i,j) \in S} (\mu_{ij}) \text{ is selected and entered into the basis.}$$

The dual simplex iteration is performed. This is repeated until

$S' = \{\emptyset\}$ . Since the value of  $\mu$  is gradually increasing and  $C$  is gradually decreasing with the increase in  $D$ , the curve  $B_3$  is a piecewise concave function.

#### Tracing $B_4$ Curve:

$B_4$  curve is a piecewise linear convex function between cost and budget. This curve is obtained by parametrically solving the following problem:

$$\text{Min. } Z = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \quad \forall (i,j) \in A$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} \geq D$$

The optimality conditions are.

$$(i) \quad \bar{c}_{pq} - \mu \bar{d}_{pq} = 0$$

$$(ii) \quad \bar{c}_{ij} - \mu \bar{d}_{ij} > 0 \quad \forall (i,j) \in L \cup U$$

$$(iii) \quad \mu \left( \sum_{(i,j) \in A} d_{ij} x_{ij} - D \right) = 0$$

The non-basic arcs are classified as:

$$(i) \quad \text{Active arcs } S' : \{(i,j) \in L \cup U : \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} > 0\}$$

$$(ii) \quad \text{Critical arcs } S'' : \{(i,j) \in L \cup U : \bar{c}_{ij} \leq 0 \text{ and } \bar{d}_{ij} > 0\}$$

(iii) Passive arcs  $S''$ :  $\{(i,j) \in L \cup U : \bar{d}_{ij} \leq 0\}$

Initially  $\sum_{(i,j) \in A} c_{ij} x_{ij}$  is minimized and let  $(C_{\min}, D)$  be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc  $(p,q)$  such that  $u_{pq} = \min_{(i,j) \in S'} (u_{ij})$  is selected and entered into the basis. The dual simplex iteration is performed. This is repeated until  $S' = \{\emptyset\}$ . A dual simplex iteration results in an extreme solution. Since the value of  $u$  is gradually increasing and both  $C$  and  $D$  are increasing, the curve  $B_4$  is a piecewise convex function.

The curves  $B_1$  and  $B_4$  together constitute a convex curve as the starting solution is same and the cost is gradually increasing. The curves  $B_2$  and  $B_3$  together constitute a concave curve as the starting solution is the same and the cost is gradually decreasing. It follows from the above discussion that the terminating solution of  $B_1$  is the starting solution for  $B_2$  and the terminating solution of  $B_2$  is the starting solution for  $B_3$ . Similarly the terminating solution of  $B_3$  is the starting solution for  $B_4$  and the terminating solution of  $B_4$  is the starting solution for  $B_1$ . Thus we obtain a closed region enclosed by the curves  $B_1 \cup B_2 \cup B_3 \cup B_4$ . Since  $B_1 \cup B_4$  is a convex curve and  $B_2 \cup B_3$  is a concave curve, the closed feasible region is a closed convex region.

### 3.4 Mathematical Formulation of the Weighted Goal Network Flow Problem:

In this section, we give the mathematical formulation of the weighted goal network flow problem.

In the network  $G = (N, A)$ , associated with each arc  $(i, j) \in A$  are two numbers  $c_{ij}$  and  $d_{ij}$ . The capacity of each arc  $(i, j) \in A$  is  $b_{ij}$ . The amount that is to be shipped from source 1 to sink  $n$  is  $V$ .

$$\text{Minimize } Z = w_1\alpha_1 + r_1\beta_1 + w_2\alpha_2 + r_2\beta_2 \quad (3.1)$$

subject to the following constraints:

#### Flow Conservation Constraints:

These constraints essentially represent the fact that flow of the commodity is conserved at all nodes, except at source and sink.

$$\sum_{(j,i) \in I(i)} x_{ji} - \sum_{(i,j) \in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (3.2)$$

#### Capacity Constraints:

These constraints essentially represent the fact that flow over an arc  $(i, j)$  cannot exceed its capacity

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall (i, j) \in A \quad (3.3)$$



### Goal Constraints:

An objective in conjunction with an aspiration level and deviations is known as goal constraint.

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1 \quad (3.4)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1 \quad (3.5)$$

### 3.5 Development of the Algorithm:

We shall now discuss, the various steps involved in finding an optimum solution for a specified  $(C_1, D_1, w_1, r_1, w_2, r_2)$ . Let us consider  $(C_1, D_1) \in R_2$  in order to trace all the possible steps of the WGNF algorithm.

Initially, the minimum cost flow problem with  $\sum_{(i,j) \in A} c_{ij} x_{ij}$  as an objective function is solved. Let  $(C_{\min}, D')$  be the minimum cost and budget obtained and it is represented by I in Fig. 3.3. The deviations axes from aspiration levels  $(C_1, D_1)$  are shown in Fig. 3.2. If the algorithm traverses in quadrant I then it minimizes  $\alpha_1$  and  $\alpha_2$  and the other two deviations are zero. Similarly  $\alpha_1$  and  $\beta_2$  are minimized in quadrant II,  $\beta_2$  and  $\beta_1$  are minimized in quadrant III and  $\alpha_2$  and  $\beta_1$  are minimized in quadrant IV. Therefore, the algorithm always minimizes only two deviations and the other two being at zero. The above statement is supported by theorems 1 to 5 given below.

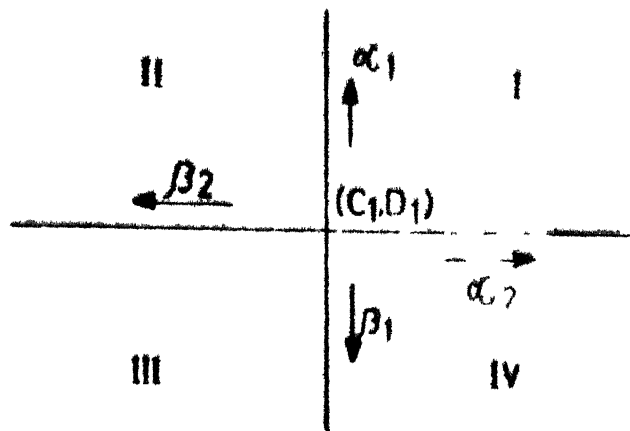


Fig. 3.2 Aspiration levels and deviations for WGNF problem

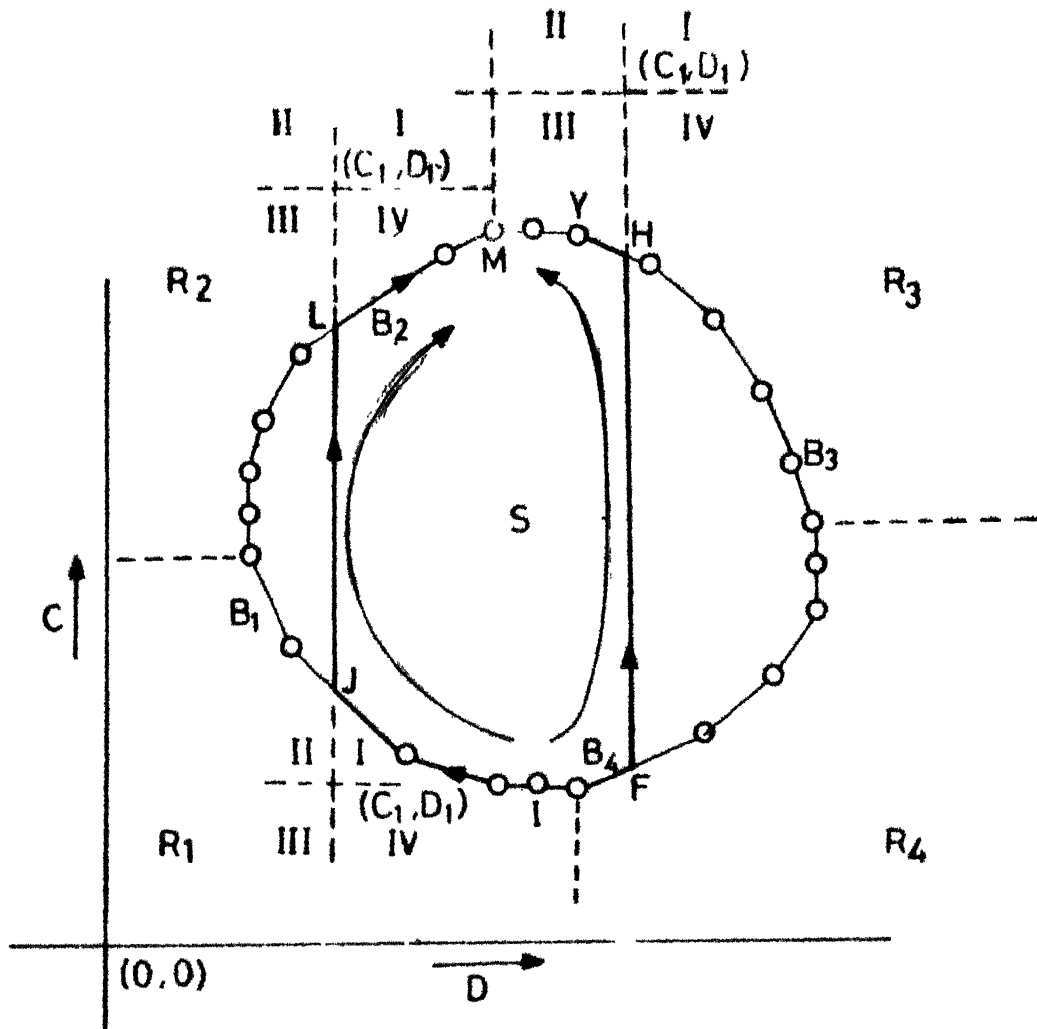


Fig.3.3 Paths traced by WGNF regorithm to obtain optimum solutions

THEOREM 1:  $\alpha_i \beta_i = 0, \forall i = 1, 2$

Proof: We will prove this theorem by contradiction.

Suppose  $(X^*, \alpha_1^*, \beta_1^*)$  be an optimum solution and  $\alpha_1^*, \beta_1^* > 0$ .

$$\text{Let } C^* = \sum_{(i,j) \in A} c_{ij} x_{ij}^*$$

The value of the objective function is

$$Z_1 = w_1 \alpha_1^* + r_1 \beta_1^* \quad (3.6)$$

The goal constraint is

$$C^* + \beta_1 - \alpha_1 = C_1 \quad (3.7)$$

$$\text{Let } \Delta = \min. (\beta_1^*, \alpha_1^*)$$

Consider the solution  $(X^*, \beta_1', \alpha_1')$ ,

$$\text{let, } \beta_1' = \beta_1^* - \Delta \quad (3.8)$$

$$\alpha_1' = \alpha_1^* - \Delta \quad (3.9)$$

substituting (3.8) and (3.9) in (3.6) and (3.7), we get,

$$\begin{aligned} Z_2 &= w_1 (\alpha_1' + \Delta) + r_1 (\beta_1' + \Delta) \\ &= w_1 \alpha_1' + r_1 \beta_1' + \Delta (w_1 + r_1) \end{aligned} \quad (3.10)$$

goal constraint is

$$C^* + \beta_1' - \alpha_1' = C_1 \quad (3.11)$$

Hence the goal constraint is satisfied.

Since  $\Delta, w_1, r_1 > 0$  it follows that  $Z_2 > Z_1$ . This contradicts that  $(X^*, \beta_1^*, \alpha_1^*)$  is an optimum solution. Therefore  $\Delta = 0$ . Hence the theorem follows.

Similarly it can be proved for  $\alpha_2 \beta_2 = 0$ . Therefore atleast two of  $\alpha_1, \beta_1, \alpha_2$  and  $\beta_2$  are zero.

QED

**THEOREM 2:** If  $(C_1, D_1) \in R_1$ , then  $(C^*, D^*) \in B_1$  and  $D^* \geq D_1$  and  $C^* \geq C_1$ .

**Proof:** Consider  $(C', D') \in S_2$ . There exists  $(C', D) \in S_1$ , such that  $D < D'$ . Since  $D < D'$ , it follows that  $\alpha_2 < \alpha'_2$  resulting in  $Z < Z'$  where  $Z$  is the value of objective function  $(C', D)$ . Hence a better solution exists in  $S_1$  for every solution in  $S_2$ .

Consider a feasible solution  $(C', D') \in S_1$ . A feasible solution  $(C, D') \in B_1$  exists such that  $C < C'$ . Since  $C < C'$  it follows that  $\alpha_1 < \alpha'_1$ . It results in  $Z < Z'$  where  $Z$  and  $Z'$  are objective functions of  $(C, D')$  and  $(C', D')$  respectively. Therefore for any solution in  $S$ , a better solution exists on  $B_1$ .

Let  $(C', D') \in B_1$  and  $D' < D_1$ . There exists  $(C, D_1) \in B_1$  and  $C < C'$ . Therefore  $\alpha_1 < \alpha'_1$  resulting in  $Z < Z'$ .

Let  $(C'', D'') \in B_1$  and  $C'' < C_1$ . There exists  $(C_1, D)$  and  $C_1 > C''$ . Therefore  $\alpha_2 < \alpha_2''$  and  $\beta_1 < \beta_1''$ , resulting  $Z < Z''$ . Hence an optimum solution is  $(C^*, D^*) \in B_1$  and  $D^* \geq D$ , and  $C^* \geq C_1$ . It follows from theorem 1 and above that if  $(C_1, D_1) \in R_1$  then the optimum solution is  $(X^*, \alpha_1^*, \alpha_2^*)$  and  $\beta_1, \beta_2 = 0$ .

Similarly we can prove the following theorems.

QED

THEOREM 3: If  $(C_1, D_1) \in R_2$  then  $(C^*, D^*) \in B_2$  and  $D^* \geq D_1$  and  $C^* \leq C_1$ .

The optimum solution is  $(X^*, \alpha_2^*, \beta_1^*)$  and  $\alpha_1, \beta_2 = 0$ .

THEOREM 4: If  $(C_1, D_1) \in R_3$ , then  $(C^*, D^*) \in B_3$  and  $D^* \leq D_1$  and  $C^* \leq C_1$ .

The optimum solution is  $(X^*, \beta_1^*, \beta_2^*)$  and  $\alpha_1, \alpha_2 = 0$ .

THEOREM 5: If  $(C_1, D_1) \in R_4$ , then  $(C^*, D^*) \in B_4$  and  $D^* \leq D_1$  and  $C^* \geq C_1$ .

The optimum solution is  $(X^*, \beta_2^*, \alpha_1^*)$  and  $\alpha_2, \beta_1 = 0$ .

QED

If  $D_1 \leq D'$  then the algorithm moves along the  $B_1$  curve in the direction of  $J$  as shown in Fig. 3.3. Otherwise it traverses along  $B_4$ . The algorithm evaluates the objective function values of extreme-solutions and two non-extreme solutions, namely  $(C, D_1) \in B_1 \cup B_2 \cup B_3 \cup B_4$  and  $(C_1, D) \in B_1 \cup B_2 \cup B_3 \cup B_4$ , because one of these solutions is optimum if  $(C_1, D_1) \notin S$ . We ascertain the above statement in theorems 6 to 9 given below.

THEOREM 6: If  $(C_1, D_1) \notin S$ , then one of the extreme solutions of  $B_1 \cup B_2 \cup B_3 \cup B_4$  is optimum.

Proof: Consider two adjacent extreme solutions  $(C', D')$  and  $(C'', D'')$ . Any non-extreme solution  $(\bar{C}, \bar{D})$  lying between  $(C', D')$  and  $(C'', D'')$  can be expressed as convex combination of

$(C', D')$  and  $(C'', D'')$ . Therefore  $Z'$  or  $Z''$  is atleast as good as any  $\bar{Z}$ , depending on  $Z' \leq Z''$  or  $Z'' \leq Z'$ . Hence the theorem follows.

QED

**THEOREM 7:** If  $(C_1, D) \in B_1$  lies between two adjacent extreme solutions  $(C', D') \in B_1$  and  $(C'', D'') \in B_1$  such that  $C_1 > C''$ ,  $D' > D_1$  and satisfying  $Z' \geq Z$  where  $Z'$ ,  $Z$  and  $Z''$  are objective function values of  $(C', D')$ ,  $(C_1, D)$  and  $(C'', D'')$  respectively, then  $(C_1, D)$  is optimum.

$$\text{Proof: } Z'' = w_2 \alpha_2'' + r_1 \beta_1'' \quad (3.12)$$

$$Z = w_2 \alpha_2$$

Since  $\alpha_2 < \alpha_2''$ , it follows that  $Z < Z''$ .  $(C_1, D)$  is atleast as good as any other solution lying between  $(C_1, D)$  and  $(C', D')$  as they can be expressed as a convex combination of  $(C_1, D)$  and  $(C', D')$  and  $Z' \geq Z$ . Similarly  $(C_1, D)$  is better than any other solution lying between  $(C_1, D)$  and  $(C'', D'')$  because they can be expressed as a convex combination of  $(C_1, D)$  and  $(C'', D'')$  and  $Z'' > Z$ . According to theorem 2, for any solution in  $S$ , a better solution exists on  $B_1$ . Hence  $(C_1, D)$  is atleast as good as any neighbouring solution. Therefore  $(C_1, D)$  is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of  $(C_1, D) \in B_4$ ,  $(C, D_1) \in B_2$  and  $(C, D_1) \in B_3$  lying between two adjacent extreme solutions,  $(C', D')$  and  $(C'', D'')$  satisfying  $Z' \geq Z$  then they are optimum.

QED

**THEOREM 8:** If  $(C, D_1) \in B_1$  lies between two adjacent extreme solutions  $(C', D') \in B_1$  and  $(C'', D'') \in B_1$  such that  $C > C''$ ,  $C'' > C_1$  and satisfying  $Z < Z''$  where  $Z, Z', Z''$  are objective function values of  $(C, D_1)$ ,  $(C', D')$  and  $(C'', D'')$  respectively, then  $(C, D_1)$  is optimum.

$$\begin{aligned}\text{Proof: } Z' &= w_1 \alpha_1' + r_2 \beta_2' \\ Z &= w_1 \alpha_1\end{aligned}$$

Since  $\alpha_1 < \alpha_1'$ , it follows that  $Z < Z'$ .  $(C, D_1)$  is a better than any solution between  $(C', D')$  and  $(C, D_1)$  and between  $(C, D_1)$  and  $(C'', D'')$  as they can be expressed as convex combination of these solutions and  $Z < Z'$  and  $Z < Z''$ . According to theorem 2, for any solution in  $S$ , a better solution exists on  $B_1$ . Hence  $(C, D_1)$  is atleast as good as any neighbouring solution. Therefore  $(C, D_1)$  is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of  $(C, D_1) \in B_4$ ,  $(C_1, D) \in B_2$  and  $(C_1, D) \in B_3$  lying between two

adjacent extreme solutions,  $(C', D')$  and  $(C'', D'')$  satisfying  $Z < Z''$  then they are optimum.

QED

THEOREM 9: If  $(C, D_1) \in B_1$  and  $(C_1, D) \in B_1$  lie between two adjacent extreme solutions  $(C', D') \in B_1$  and  $(C'', D'') \in B_1$  such that  $C_1 > C''$ , then  
 (i)  $(C_1, D)$  is optimum if  $\underline{Z} \leq \bar{Z}'$  and  
 (ii)  $(C, D_1)$  is optimum if  $\underline{Z} \geq \bar{Z}$   
 where  $\underline{Z}$  and  $\bar{Z}$  are the objective function values of  $(C_1, D)$  and  $(C, D_1)$ .

Proof: We shall prove case (i).

Let  $Z'$  and  $Z''$  be objective function values of  $(C', D')$  and  $(C'', D'')$

$$Z'' = w_2 \alpha_2'' + r_1 \alpha_1''$$

$$\underline{Z} = w_2 \alpha_2$$

Since  $\alpha_2 < \alpha_2''$ , it follows that  $\underline{Z} < Z''$ .  $(C_1, D)$  is at least as good as any solution lying between  $(C_1, D)$  and  $(C, D_1)$  because they can be expressed as convex combination of  $(C_1, D)$  and  $(C, D_1)$  and  $\underline{Z} \leq \bar{Z}$ . Similarly  $(C_1, D)$  is better than any solution lying between  $(C_1, D)$  and  $(C'', D'')$  as they can be expressed as convex combination of  $(C_1, D)$  and  $(C'', D'')$  and  $\underline{Z} < Z''$ . According to theorem 2, for any solution in  $S$ , there exists a better solution on  $B_1$ . Hence  $(C_1, D)$  is at least as



good as any neighbouring solution. Therefore  $(C_1, D)$  is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Case (ii) is proved as follows:

$$Z' = w_1 \alpha_1' + r_2 \beta_2'$$

$$\bar{Z} = w_1 \bar{\alpha}_1$$

Since  $\bar{\alpha}_1 < \alpha_1'$ , it follows that  $\bar{Z} < Z'$  using theorem 2,  $\bar{Z} < Z'$  and  $\bar{Z} \leq Z$ , it can be proved that  $(C, D_1)$  is a local optima, which is also a global optima in linear programming. Hence  $(C, D_1)$  is an optimum solution.

Similar results can be proved for the cases when  $(C, D_1) \in B_2 \cup B_3 \cup B_4$  and  $(C_1, D) \in B_2 \cup B_3 \cup B_4$  lie between two adjacent extreme solutions then one of them is optimum depending on  $\bar{Z} \geq Z$  or  $\bar{Z} < Z$ .

QED

We can infer from theorems (7) to (9) that the non-extreme solutions  $(C, D_1)$  and  $(C_1, D)$  behave as extreme solutions and hence the algorithm evaluates the objective function values of these non-extreme solutions.

If  $Z$  corresponding to  $(C, D)$  the current solution is greater than or equal to  $Z''$  of  $(C'', D'')$  the previous extreme solution, then  $(C'', D'')$  is optimum and the algorithm stops. This is proved in the following theorem.

THEOREM 10: If  $(C_1, D_1) \in R_1$  and  $(C', D') \in B_1$ ,  $(\bar{C}, \bar{D}) \in B_1$  and  $(C'', D'') \in B_1$  be three adjacent extreme solutions satisfying  $Z'' \geq \bar{Z}$  and  $Z' \geq \bar{Z}$  then  $(\bar{C}, \bar{D})$  is an optimum solution.

Proof:  $(\bar{C}, \bar{D})$  is atleast as good as any other solution lying between  $(\bar{C}, \bar{D})$  and  $(C', D')$  as they can be expressed as convex combination of these two solutions and  $Z' \geq \bar{Z}$ . Similarly  $(\bar{C}, \bar{D})$  is atleast as good as any other solution lying between  $(\bar{C}, \bar{D})$  and  $(C'', D'')$  because they can be expressed as convex combination of these two solutions and  $Z'' \geq \bar{Z}$ . According to theorem 2, if  $(C_1, D_1) \in R_1$  then a better solution exists on  $B_1$  than any solution in  $S$ . Therefore  $(\bar{C}, \bar{D})$  is a local optima, as it is better than any neighbouring solution. In linear programming local optima is also global optima and hence  $(\bar{C}, \bar{D})$  is optimum. Similarly, this type of result can be proved for the cases,  $(C_1, D_1) \in R_2 \cup R_3 \cup R_4$ .

QED

If  $D_1 \leq D_{\min}$ , then the algorithm moves along  $B_1 \cup B_2$  and obtains an optimum solution. Otherwise the algorithm halts at  $(C, D_1) \in B_1$  represented by  $J$  in Fig. 3.3. If  $(C_1, D_1) \in R_1$  then  $(C, D_1)$  is an optimum solution which follows from the theorem 8. Otherwise it moves vertically up along  $JL$  in the direction of  $C_1$ .

We will discuss in detail how the algorithm increases the total cost  $C$ , while maintaining the total budget  $D$  constant. This is achieved by simultaneously augmenting the flow in two cycles.

An arc  $(i,j)$  is a forward arc if the flow is sent in the direction of the arc, and a backward arc if the flow is sent against the direction of the arc. We will refer to this as orientation of the arc. The basis structure of current solution is  $(T \cup \{(p,q)\}, L, U)$  and its basis consists of exactly one cycle formed by the arc  $(p,q)$ . This cycle is denoted by  $W_{pq}$ . Now, define a set

$$S^0 = \{(i,j) \in L \cup U: \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and} \\ (\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{ij}} \times \bar{c}_{ij} > 0) \}.$$

The above conditions are derived as follows.

#### Criteria for Selecting a Non-Basic Arc for 2 Cycle Flow Augmentation:

We will derive the conditions that the non-basic arc  $(r, s) \in L \cup U$  must satisfy, such that by simultaneously augmenting flow in cycles  $W_{pq}$  and  $W_{rs}$ , the total cost increases and the total budget remains constant.

The basis structure is  $(T \cup \{(p,q)\}, L, U)$ .

Let  $\Delta_{pq}$  be the flow to be circulated in cycle  $W_{pq}$ .

Let  $\Delta_{rs}$  be the flow to be circulated in cycle  $W_{rs}$ .

For the total budget  $D$  to remain constant,

$$\Delta_{pq} \bar{d}_{pq} + \Delta_{rs} \bar{d}_{rs} = 0 \quad (3.13)$$

In order to increase the total cost  $C$ ,

$$\Delta_{pq} \bar{c}_{pq} + \Delta_{rs} \bar{c}_{rs} > 0 \quad (3.14)$$

Using (3.13),

$$\frac{\Delta_{rs}}{\Delta_{pq}} = - \left( \frac{\bar{d}_{pq}}{\bar{d}_{rs}} \right) = K \quad (3.15)$$

Since  $\Delta_{pq}, \Delta_{rs} > 0$ , hence in order to satisfy (3.13), we get,

$$\frac{\bar{d}_{pq}}{\bar{d}_{rs}} < 0 \quad (3.16)$$

Dividing (3.14) by  $\Delta_{pq}$  and substituting (3.15), we get,

$$\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{rs}} \times \bar{c}_{rs} > 0 \quad (3.17)$$

Hence if an arc  $(r, s)$  satisfies (3.16) and (3.17) and augmenting flow  $\Delta_{pq}$  and  $\Delta_{rs} K$  in cycles  $W_{pq}$  and  $W_{rs}$ , the total cost increases and total budget  $D$  remains constant.

If  $S^0 = \{\emptyset\}$  then reverse the orientation of the arc  $(p, q)$  and set  $\bar{c}_{pq} = -\bar{c}_{pq}$ ,  $\bar{d}_{pq} = -\bar{d}_{pq}$ . Then, we again find  $S^0$ . If  $S^0$

is not empty, select an arc  $(r,s) \in S^0$  and enter it into the basis. This results in the formation of two cycles  $W_{pq}$  and  $W_{rs}$ . The orientation of a cycle  $W_{ij}$ ,  $(i,j) \in L \cup U$  is same as that of the arc  $(i,j)$ . Let  $\bar{W}_{ij}$  and  $\underline{W}_{ij}$  be the sets of arcs, along and opposite to the orientation of the cycle  $W_{ij}$ , respectively. Let  $\Delta_{pq}$  be the amount of flow to be augmented in  $W_{pq}$  and  $K \cdot \Delta_{pq}$  be the flow to be augmented in cycle  $W_{rs}$  where  $K = -\bar{d}_{pq}/\bar{d}_{rs}$ . Next, we determine  $\Delta_{pq}$  as follows.

Consider an arc  $(i,j) \in (W_{pq} \cap W_{rs})$ . The flow in arc  $(i,j)$  after augmenting flow in  $W_{pq}$  and  $W_{rs}$  is,

$$\begin{aligned}\bar{x}_{ij} &= x_{ij} + \Delta_{pq} I_1 + I_2 K \Delta_{pq} \\ &= x_{ij} + \Delta_{pq} (I_1 + I_2 K)\end{aligned}$$

where,

$$x_{ij} = \text{existing flow in } (i,j)$$

$$I_1 = +1, \text{ if } (i,j) \in \bar{W}_{pq}$$

$$I_1 = -1, \text{ if } (i,j) \in \underline{W}_{pq}$$

$$I_2 = +1, \text{ if } (i,j) \in \bar{W}_{rs}$$

$$I_2 = -1, \text{ if } (i,j) \in \underline{W}_{rs}$$

$$\Delta_{pq} (I_1 + I_2 K) = \text{resultant augmented flow in arc } (i,j).$$

In order to satisfy the capacity constraints,

$$0 \leq x_{ij} + \Delta_{pq} (I_1 + I_2 K) \leq b_{ij}$$

If  $(I_1 + I_2 K) > 0$ , then

$$\Delta_{pq} \leq \frac{(b_{ij} - x_{ij})}{(I_1 + I_2 K)}$$

If  $(I_1 + I_2 K) < 0$ , then

$$\Delta_{pq} \leq - \frac{x_{ij}}{(I_1 + I_2 K)}$$

In this manner  $\Delta_{pq}$  is calculated for all arcs  $(i,j) \in (W_{pq} \cup W_{rs})$ . Therefore,

$$\Delta_{pq} = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left\{ - \frac{x_{ij}}{(I_1 + I_2 K)} \text{ if } (I_1 + I_2 K) < 0, \right. \\ \left. \frac{(b_{ij} - x_{ij})}{(I_1 + I_2 K)} \text{ if } I_1 + I_2 K > 0 \right\}$$

$$\Delta_{pq} = \min. \left( \Delta_{pq}, \frac{(C_1 - C)}{(\bar{c}_{pq} + K\bar{c}_{rs})} \right)$$

Once  $\Delta_{pq}$  is obtained, the flows are updated in  $W_{pq}$  and  $W_{rs}$  cycles. One of the arcs  $(u,v) \in (W_{pq} \cup W_{rs})$  leaves at one of its bounds respectively. The dual variables,  $c_{ij}$ 's,  $d_{ij}$ 's,  $\forall (i,j) \in L \cup U$ . C, D and Z are updated respectively.

Arc  $(p,q)$  for the next iteration is obtained as follows:

- (a) If  $(u,v) = (i,j) \in (W_{pq})$ , set  $(p,q) = (r,s)$ .
- (b) If  $(u,v) = (i,j) \in (\{W_{pq} \cap W_{rs}\} \cup W_{rs})$ ,  $(p,q)$  does not change.

If the arc  $(p,q)$  is at one of its bounds, reverse the orientation of  $(p,q)$  and set  $\bar{c}_{pq} = -\bar{c}_{pq}$ ,  $\bar{d}_{pq} = -\bar{d}_{pq}$ . By selecting an arc  $(r,s) \in S^0$  at each iteration and augmenting flows in  $W_{pq}$  and  $W_{rs}$ , the algorithm moves along  $J, L$  shown in Fig. 3.3. If  $(C_1, D_1) \in S$ , then the algorithm obtains this solution which is an optimum solution and hence stops. If  $S^0$  is empty then the current solution,

$$(C, D) \in B_2 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$$

or,

$$(C, D) \in B_3 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} < 0$$

according to the theorem 11.  $L$  in Fig. 3.3 represents this solution.

THEOREM 11: If  $S^0 = \{\emptyset\}$  then the current solution  $(C, D) \in B_2$  if  $\bar{c}_{pq}/\bar{d}_{pq} > 0$  and  $(C, D) \in B_3$  if  $\bar{c}_{pq}/\bar{d}_{pq} < 0$ .

Proof: The necessary and sufficient conditions for  $(C, D) \in B_2$  are, the non-basic arcs  $(i,j) \in L \cup U$  whose  $\bar{c}_{ij} > 0$  and  $\bar{d}_{ij} < 0$ , do not exist.

The necessary and sufficient conditions for  $(C, D) \in B_3$  are, the non-basic arcs  $(i,j) \in L \cup U$  whose  $\bar{c}_{ij} > 0$  and  $\bar{d}_{ij} > 0$ , do not exist.

$$\text{Suppose,} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$$

Arcs of the type  $\bar{c}_{ij} > 0$  and  $\bar{d}_{ij} < 0$  satisfy (3.16) and (3.17) if  $\bar{c}_{pq} > 0$  and  $\bar{d}_{pq} > 0$ . Arcs of the type  $\bar{c}_{ij} > 0$  and  $\bar{d}_{ij} = 0$  satisfy (3.16) and (3.17), if  $\bar{c}_{pq} < 0$  and  $\bar{d}_{pq} < 0$ .

Since  $S^0 = \{\emptyset\}$ , the non-basic arcs  $(i,j)$  of the type  $\bar{c}_{ij} \geq 0$  and  $\bar{d}_{ij} < 0$  are absent. Therefore  $(C, D) \in B_2$ .

Suppose,  $\bar{c}_{pq}/\bar{d}_{pq} < 0$

Arcs of the type  $\bar{c}_{ij} \geq 0$  and  $\bar{d}_{ij} > 0$  satisfy  $\bar{d}_{pq} < 0$  and  $\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{ij}} \times \bar{c}_{ij} > 0$  if  $\bar{c}_{pq} > 0$  and  $\bar{d}_{pq} < 0$ . Since  $S^0 = \{\emptyset\}$  the non-basic arcs  $(i,j)$  whose  $\bar{c}_{ij} > 0$  and  $\bar{d}_{ij} > 0$ ,  $\forall (i,j) \in L \cup U$  are absent. Hence  $(C, D) \in B_3$ .

QED

Now, the algorithm moves towards M along the curve  $B_2$ . According to theorem 2, if  $(C_1, D_1) \in R_2$  the  $(C^*, D^*) \in B_2$  and  $D^* \geq D_1$ , and  $C^* \leq C_1$ , it is justified in moving along L, M for finding an optimum solution.

Similarly if  $(C_1, D_1) \in R_3$ , the algorithm traverses along I, F, H, Y and finds an optimum solution. If  $(C_1, D_1) \in R_3$  and  $D_1 \geq D_{\max}$ , then the algorithm moves along  $B_4$  and  $B_3$  curves, and obtains an optimum solution.



### 3.6 Statement of Algorithm:

A formal statement of the algorithm organised in a manner suitable for computer implementation is given below:

Step.1: Set count to zero. Solve the minimum cost flow problem

with  $\sum_{(i,j) \in A} c_{ij} x_{ij}$  as the objective function value.

Let  $(B,L,U)$  be the optimum basis structure obtained

and  $X$  be the solution. Compute,

$$C = \sum_{(i,j) \in A} c_{ij} x_{ij}, \text{ and } D = \sum_{(i,j) \in A} d_{ij} x_{ij}.$$

$(C, D)$  is the minimum cost and budget respectively.

Treating  $B$  as the basic tree, define the dual variables

$\pi_j^c$ 's and  $\pi_j^d$ 's as follows.

$$\pi_1^c = 0, \quad \pi_j^c = \pi_1^c + c_{1j}, \quad \forall (i,j) \in E$$

$$\pi_1^d = 0, \quad \pi_j^d = \pi_1^d + d_{1j}, \quad \forall (i,j) \in B$$

Define  $\bar{c}_{ij}$  and  $\bar{d}_{ij}$  for all non-basic arcs  $(i,j) \in L \cup U$  as follows:

$$\bar{c}_{ij} = \begin{cases} \pi_i^c - \pi_j^c + c_{ij}, & \forall (i,j) \in L \\ \pi_j^c - \pi_1^c - c_{1j}, & \forall (i,j) \in U \end{cases}$$

$$\bar{d}_{ij} = \begin{cases} \pi_1^d - \pi_j^d + d_{1j}, & \forall (i,j) \in L \\ \pi_j^d - \pi_1^d - d_{1j}, & \forall (i,j) \in U \end{cases}$$

$$\text{Compute, } \alpha_1 = (C - C_1)$$

$$\alpha_2 = (D - D_1)$$

$$\beta_1 = (C_1 - C)$$

$$\beta_2 = (D_1 - D)$$

$$\alpha_i = \begin{cases} \alpha_i, & \text{if } \alpha_i > 0, \quad \forall i = 1, 2 \\ 0, & \text{if } \alpha_i < 0, \end{cases}$$

$$\beta_i = \begin{cases} \beta_i, & \text{if } \beta_i > 0, \quad \forall i = 1, 2 \\ 0, & \text{if } \beta_i < 0, \end{cases}$$

$$Z = w_1 \alpha_1 + w_2 \alpha_2 + r_1 \beta_1 + r_2 \beta_2$$

If  $D_1 < D$  then flag = -1, else flag = 1.

Set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$ ,  $D' = D$  and go to Step 2.

Step 2: Let  $S' = \{(i, j) \in L \cup U : c_{ij} \leq 0 \text{ and flag} * d_{ij} > 0\}$

If  $S' = \emptyset$  then go to Step 3. Otherwise select an arc  $(p, q) \in S'$  and enter it into the basis. Perform pivot iteration, up date  $x_{ij}$ 's.

If  $\text{flag} * D > \text{flag} * D_1$ , then circulate  $\Delta$  flow against the orientation of  $W_{pq}$  where  $\Delta = \frac{D - D_1}{d_{pq}}$ , and go to Step 7.

If  $Z > Z'$  then  $X'$  is an optimum solution, Stop, otherwise, update  $(B, L, U)$ ,  $\pi_j^C$ 's,  $\pi_j^D$ 's, set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$  and  $D' = D$ . Repeat this step.

Step 3: Let  $S' = \{(i,j) \in L \cup U: c_{ij} > 0 \text{ and } \text{flag} * \bar{d}_{ij} > 0\}$ .

Define,

$$\mu_{ij} = \frac{c_{ij}}{\bar{d}_{ij}} * \text{flag}.$$

If  $S' = \emptyset$  then set  $\text{flag} = -\text{flag}$  and go to Step 4.

Otherwise select an arc  $(p,q)$ ,  $\mu_{pq} = \min_{(i,j) \in S'} (\mu_{ij})$ , and enter it into the basis. Perform the pivot iteration, update  $x_{ij}$ 's.

If  $C > C_1 > C'$  then circulate  $\Delta$  flow against the orientation of  $W_{pq}$ , where  $\Delta = (C - C_1) / \bar{c}_{pq}$ . Set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$  and  $D' = D$ . Augment  $\Delta$  flow in  $W_{pq}$  where  $\Delta = (C - C_1) / \bar{c}_{pq}$ .

If  $\text{flag} * D > \text{flag} * D_1$ , then circulate  $\Delta$  flow against the orientation of  $W_{pq}$  where  $\Delta = (D - D_1) / \bar{d}_{pq}$ , then check if  $Z > Z'$  then  $X'$  is an optimum solution, stop.

Otherwise, go to Step 7.

If  $Z > Z'$  then  $X'$  is an optimum solution, stop.

Otherwise update  $(B, L, U)$ ,  $\pi_j^{c'}$ 's,  $\pi_j^{d'}$ 's, set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$ , and  $D' = D$ . Repeat this step.

Step 4: Let  $S' = \{(i,j) \in L \cup U: \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} = 0\}$ .

If  $S' = \emptyset$  then go to Step 5. Otherwise select an arc  $(p,q) \in S'$  and enter it into the basis. Perform the pivot iteration, update  $x_{ij}$ 's.

If  $C > C_1 > C'$  then circulate  $\Delta$  flow against the orientation of  $W_{pq}$  where  $\Delta = (C - C_1)/c'_{pq}$ .  $X$  is an optimum solution, stop.

If  $Z > Z'$  then  $X'$  is an optimum solution, stop.

Otherwise, update  $(B, L, U)$ ,  $\pi_j^c$ 's,  $\pi_j^d$ 's, Set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$  and  $D' = D$ . Repeat this step.

Step 5: Let  $S' = \{(i, j) \in L \cup U : c'_{ij} > 0 \text{ and } \text{flag} * d'_{ij} > 0\}$ .

Define,  $\lambda_{ij} = c'_{ij} * \text{flag} / d'_{ij}$ .

If  $S' = \{\emptyset\}$  then  $X'$  is an optimum solution, stop.

Otherwise select an arc  $(p, q)$   $\lambda_{pq} = \max_{(i, j) \in S'} (\lambda_{ij})$  and enter it into the basis, go to Step 6.

Step 6: Perform the pivot iteration, update  $x_{ij}$ 's.

If  $C > C_1 > C'$  then circulate  $\Delta$  flow against the orientation of  $W_{pq}$  where  $\Delta = (C - C_1)/c'_{pq}$ . Then if  $Z > Z'$ ,  $X'$  is an optimum solution, stop. Otherwise  $X$  is an optimum solution, stop.

If  $Z > Z'$  then  $X'$  is an optimum solution, stop.

Otherwise update  $(B, L, U)$ ,  $\pi_j^c$ 's,  $\pi_j^d$ 's. Set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$  and  $D' = D$ . Repeat this step.

Step 7: If count = 2 then set  $Z' = Z$ ,  $X' = X$ ,  $C' = C$ ,  $D' = D$  and go to Step 8.

If  $C \leq C_1$ , then  $X$  is an optimum solution, stop.

Otherwise the basis structure is  $(T \cup \{(p, q)\}, L, U)$ .

Define  $S^0 = \{(i,j) \in L \cup U: \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and}$

$(\bar{c}_{pq} - (\bar{d}_{pq}/\bar{d}_{ij}) * \bar{c}_{ij} > 0)\}$ . If  $S^0$  is

empty then set  $\text{count} = \text{count} + 1$ ,  $\bar{c}_{pq} = -\bar{c}_{pq}$ ,

$\bar{d}_{pq} = -\bar{d}_{pq}$ ,  $\text{status} = -\text{status}$  and go to Step 7.

Otherwise, select an arc  $(r,s) \in S^0$  and enter it into the basis  $(T \cup \{p,q\})$ .

Two cycles are formed i.e.  $W_{pq}$ ,  $W_{rs}$ . Define,

$h = -\bar{d}_{pq}/\bar{d}_{rs}$ , set  $T_{ij} = 0$ ,  $\forall (i,j) \in A$ .

Define,

$$T_{ij} = \begin{cases} T_{ij} + 1, & \text{if } (i,j) \in \bar{W}_{pq} \\ T_{ij} - 1, & \text{if } (i,j) \in W_{pq} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij} + K, & \text{if } (i,j) \in \bar{W}_{rs} \\ T_{ij} - K, & \text{if } (i,j) \in W_{rs} \end{cases}$$

$$\Delta = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left( -\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \right. \\ \left. \frac{b_{ij} - x_{ij}}{T_{ij}} \right), \text{ if } T_{ij} > 0).$$

$$\Delta_{pq} = \min. \left( \Delta, \frac{(C_1 - C)}{(\bar{c}_{pq} + K * \bar{c}_{rs})} \right)$$

Update  $\Delta_{pq}$  in  $W_{pq}$  and  $K \Delta_{pq}$  in  $W_{rs}$ .

Define  $S^0 = \{(i,j) \in L \cup U: \bar{d}_{pq}/\bar{d}_{ij} < 0 \text{ and}$

$(\bar{c}_{pq} - (\bar{d}_{pq}/\bar{d}_{ij}) * \bar{c}_{ij} > 0)\}$ . If  $S^0$  is empty then set  $\text{count} = \text{count} + 1$ ,  $\bar{c}_{pq} = -\bar{c}_{pq}$ ,  $\bar{d}_{pq} = -\bar{d}_{pq}$ ,  $\text{status} = -\text{status}$  and go to Step 7. Otherwise, select an arc  $(r,s) \in S^0$  and enter it into the basis  $(T \cup \{p,q\})$ .

Two cycles are formed i.e.  $W_{pq}$ ,  $W_{rs}$ . Define,

$K = -\bar{d}_{pq}/\bar{d}_{rs}$ , set  $T_{ij} = 0$ ,  $\forall (i,j) \in A$ .

Define,

$$T_{ij} = \begin{cases} T_{ij} + 1, & \text{if } (i,j) \in \bar{W}_{pq} \\ T_{ij} - 1, & \text{if } (i,j) \in W_{pq} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij} + K, & \text{if } (i,j) \in \bar{W}_{rs} \\ T_{ij} - K, & \text{if } (i,j) \in W_{rs} \end{cases}$$

$$\Delta = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left( -\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \right. \\ \left. \frac{b_{ij} - x_{ij}}{T_{ij}} \right), \text{ if } T_{ij} > 0).$$

$$\Delta_{pq} = \min. \left( \Delta, \frac{(C_{pq} - C_{rs})}{(\bar{c}_{pq} + K * \bar{c}_{rs})} \right)$$

Update  $\Delta_{pq}$  in  $W_{pq}$  and  $K \Delta_{pq}$  in  $W_{rs}$ .

Update  $(B, L, U)$ ,  $\pi_j^c$ 's,  $\pi_j^d$ 's.

The arc  $(p, q)$  for the next iteration is obtained as follows.

Let  $(u, v)$  be the leaving arc at its respective bound.

If  $(u, v) = (i, j) \in W_{pq}$ , set  $(p, q) = (r, s)$

If  $(u, v) = (i, j) \in (W_{pq} \cap W_{rs} \cup W_{rs})$ ,  $(p, q)$  does not change

If  $x_{pq} = 0$  or  $x_{pq} = b_{pq}$ , then set  $\bar{c}_{pq} = -\bar{c}_{pq}$ ,  $\bar{d}_{pq} = -\bar{d}_{pq}$ ,  
status = - status. Repeat this step.

Step 8: The current basis structure is  $(T \cup \{(p, q)\}, L, U)$ .

If  $\bar{c}_{pq} = 0$  then the current solution  $X$  is optimum, stop.

If  $\bar{c}_{pq} < 0$ , then set  $c_{pq} = -\bar{c}_{pq}$ ,  $\bar{d}_{pq} = -\bar{d}_{pq}$ ,  
status = -status.

If  $\bar{d}_{pq} > 0$ , then set flag = 1 else set flag = -1,

Go to Step 6.

### 3.7 Numerical Example:

In this section a small-network flow problem is solved to illustrate the various steps of WGNF algorithm. The network is shown in Fig. 3.4. The numbers  $c_{ij}$ ,  $b_{ij}$ ,  $d_{ij}$  are indicated over each arc. An amount of 5 units of flow is sent from source 1 to sink 6. The aspiration levels and weights are specified as follows

$$w_1 = 0.15, w_2 = 0.10, r_1 = 0.35, r_2 = 0.4, C_1 = 165 \text{ and } D_1 = 125.$$

The steps of the algorithm are summarized in Table 3.1. The  $\uparrow$  indicates the basic pivot arc and  $\downarrow$  indicates the basic arc leaving the basis. The graphs corresponding to the basis at various iterations are shown in Fig. 3.6. In these graphs, the basic pivot arc is drawn as dashed lines. The algorithm traced the path I, J, L, N as shown in Fig. 3.5 and obtained  $C^* = 145$  and  $D^* = 155$  as an optimum solution.

### 3.8 Mathematical Formulation of Interval Goal Network Flow Problem

The mathematical formulation of interval goal network flow problem is as follows:

$$\text{Min. } Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2 \quad (3.18)$$

s.t.

$$\sum_{(j,i) \in I(1)} x_{ji} - \sum_{(i,j) \in O(1)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall i \in N \\ 0, & \text{otherwise} \end{cases} \quad (3.19)$$

$$0 < x_{ij} < b_{ij}, \quad \forall (i,j) \in A \quad (3.20)$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_1 - \alpha_1 = C_1 \quad (3.21)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_2 - \alpha_2 = D_1 \quad (3.22)$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} + \beta_3 - \alpha_3 = C_2 \quad (3.23)$$

$$\sum_{(i,j) \in A} d_{ij} x_{ij} + \beta_4 - \alpha_4 = D_2 \quad (3.24)$$



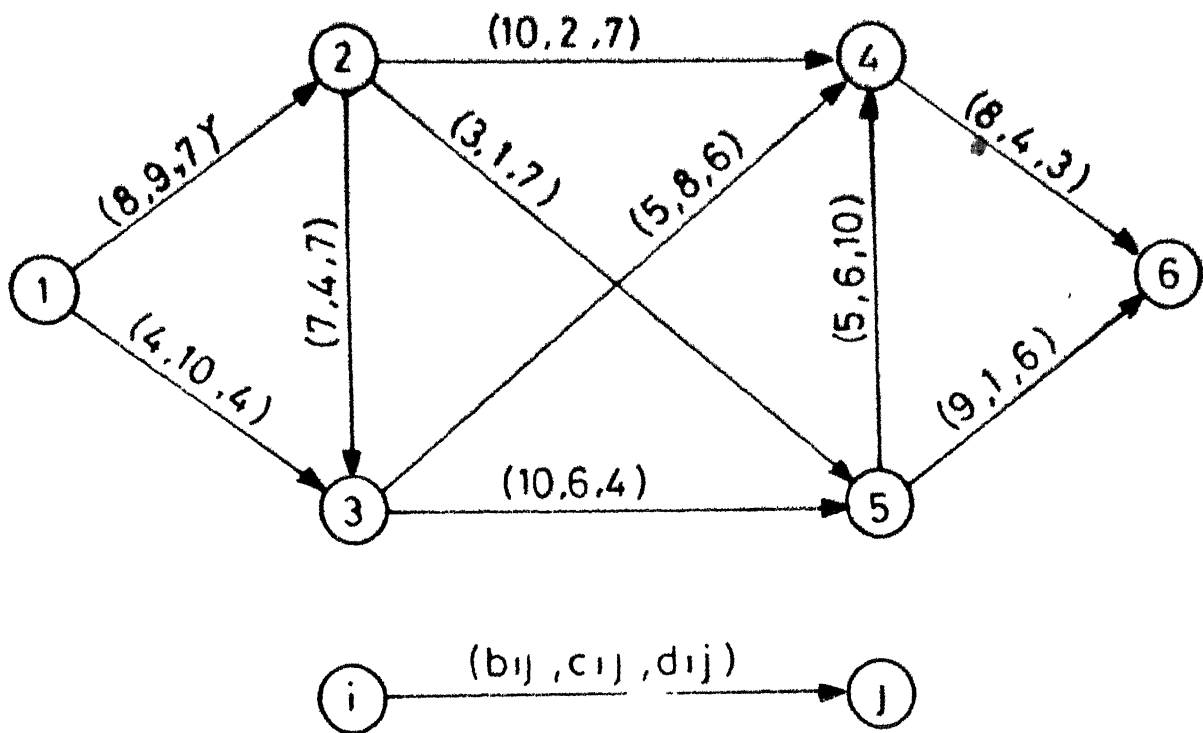


Fig. 3 4 Numerical example for the WOLF problem

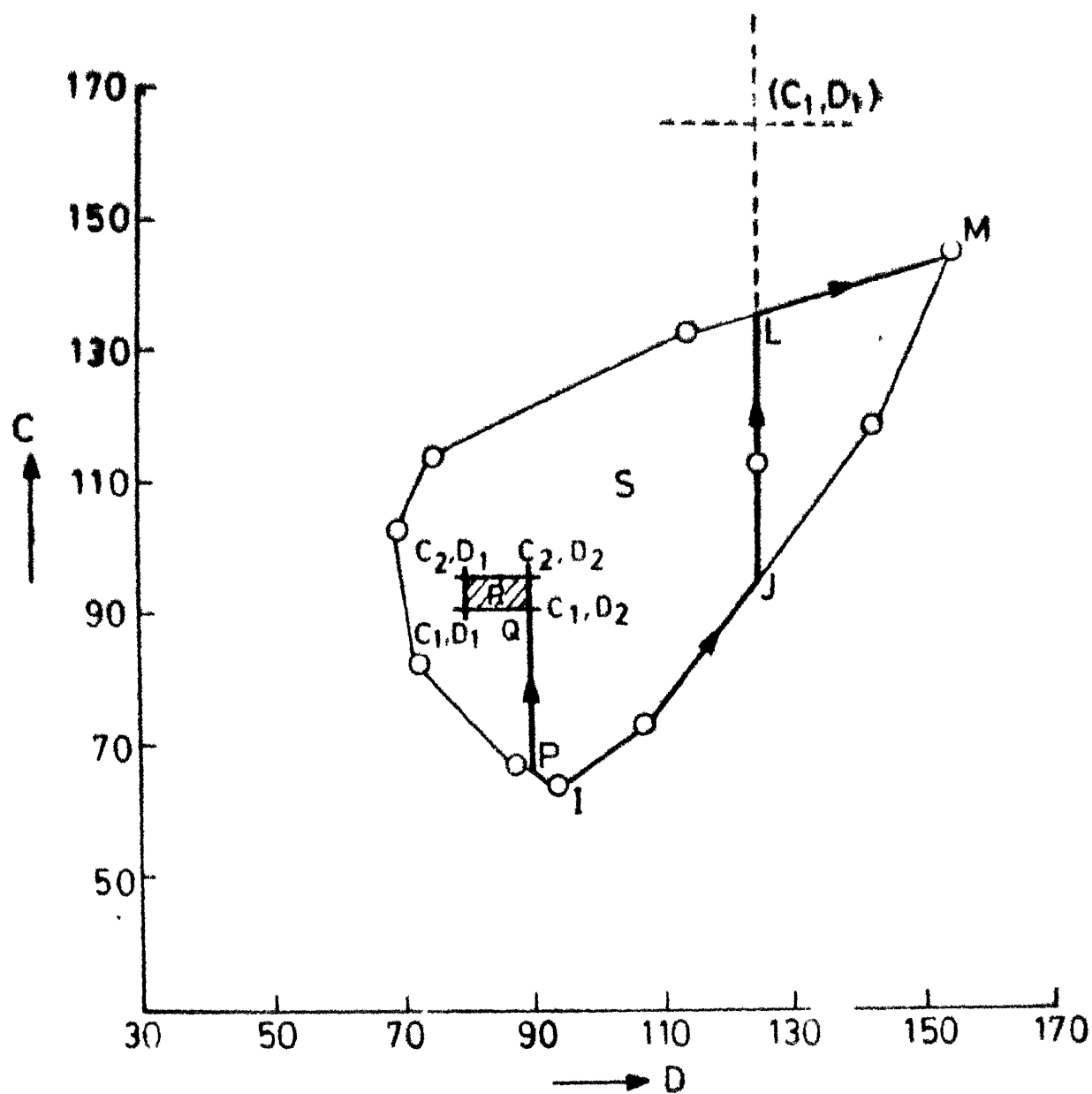


Fig. 3.5 Paths traced by WGNF algorithm and IGNF algorithm for the numerical example

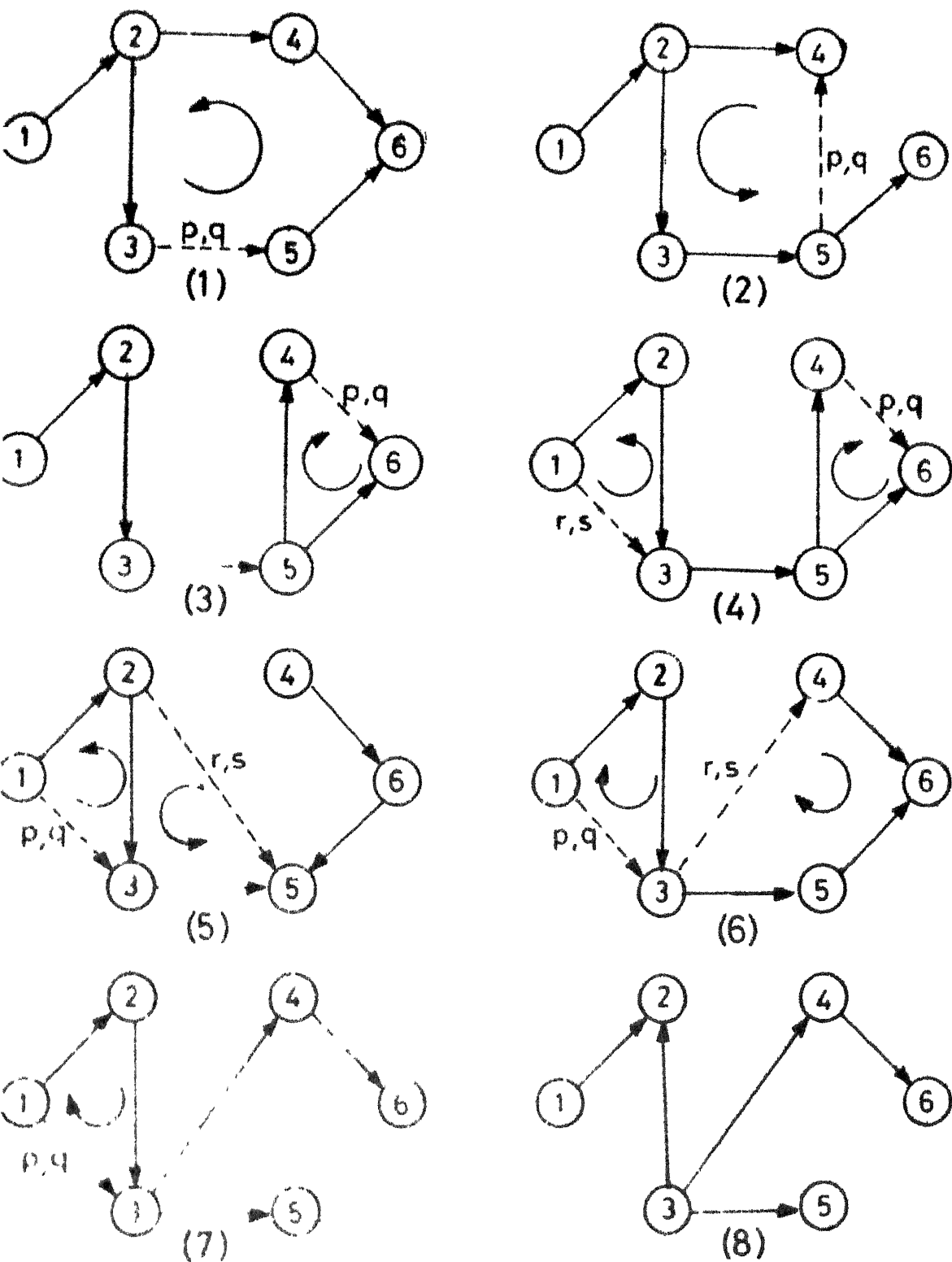


Fig 3.6 Basis in various interactions

Table 3.1: Solution of the WGNF Problem.

C D Z			Basic Tree Arcs					Non-tree Arcs				
0.63	94	38.8	$(i,j)$	$(1,2)$	$(2,4)$	$(2,3)$	$(4,6)$	$(5,6)$	$(1,3)$	$(2,5)$	$(3,4)$	$(3,5)$ $(5,4)$
			$x_{ij}$	5	2	3	2	3	0	3	0	0
			$\bar{c}_{ij}$	0	0	0	0	0	-3	4	10	5
			$\bar{d}_{ij}$	0	0	0	0	0	-10	-3	6	7 $\uparrow(p,q)$
0.373	108	33.9	$(i,j)$	$(1,2)$	$(2,4)$	$(2,3)$	$(3,5)$	$(5,6)$	$(1,3)$	$(2,5)$	$(3,4)$	$(4,6)$ $(5,4)$
			$x_{ij}$	5	0	2	2	5	0	3	0	0
			$\bar{c}_{ij}$	0	0	0	0	0	-3	9	10	-5
			$\bar{d}_{ij}$	0	0 $\downarrow$	0	0	0	-10	4	6	-7 $\uparrow(p,q)$
0.72	108	33.9	$(i,j)$	$(1,2)$	$(2,3)$	$(3,5)$	$(5,4)$	$(5,6)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,4)$ $(4,6)$
			$x_{ij}$	5	2	2	0	5	0	0	3	0
			$\bar{c}_{ij}$	0	0	0	0	0	-3	-14	9	10
			$\bar{d}_{ij}$	0	0	0	0	0	-10	-14	4	6 $\uparrow(p,q)$
1.0	94.9	125	$(i,j)$	$(1,2)$	$(2,3)$	$(3,5)$	$(5,4)$	$(5,6)$	$(1,3)$	$(2,4)$	$(2,5)$	$(3,4)$ $(4,6)$
			$x_{ij}$	5	2	2	2.43	2.57	0	0	3	0
			$\bar{c}_{ij}$	0	0	0	0	0	-3	-14	9	10
			$\bar{d}_{ij}$	0	0	0	0	0	-10 $\uparrow(r,s)$	-14	4	6 $\uparrow(p,q)$

				(1,j)	(1,2)	(2,3)	(4,6)	(3,5)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)
				$x_{ij}$	3.2	0.2	5.0	2.0	0	1.8	0	3	0	5
112.6	125	18.34		$\bar{c}_{ij}$	0	0	0	0	0	-3	-5	9	5	-9
				$\bar{d}_{ij}$	0	0	0	0	0	$\downarrow \uparrow (p,q)$ -10	-7	$\uparrow (r,s)$ 4	$\uparrow (r,s)$ -1	-1
				(i,j)	(1,2)	(2,3)	(3,5)	(4,6)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)
				$x_{ij}$	2	2	5	5	0	3	0	0	0	5
136	125	10.15		$\bar{c}_{ij}$	0	0	0	00	0	-3	-5	-9	5	-9
				$\bar{d}_{ij}$	0	0	0	0	0	$\downarrow \uparrow (p,q)$ -10	-10	-4	$\uparrow (r,s)$ -1	-1
				(i,j)	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)
				$x_{ij}$	2	2	0	5	5	3	0	0	5	0
136	125	10.15		$\bar{c}_{ij}$	0	0	0	0	0	3	-10	-9	-4	-5
				$\bar{d}_{ij}$	0	0	0	0	0	$\downarrow \uparrow (p,q)$ 10	-6	-4	-8	-1
				(i,j)	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)
143	145	10.00		$x_{ij}$	5	5	0	5	5	0	0	0	5	0
				$\bar{c}_{ij}$	0	0	0	0	0	-3	-10	-9	-4	-5
				$\bar{d}_{ij}$	0	0	0	0	0	-10	-6	-4	-8	1

where,

$C_1$  = lower limit on total cost.

$C_2$  = upper limit on total cost.

$D_1$  = lower limit on total budget.

$D_2$  = upper limit on total budget.

We are interested in a solution between  $C_1$  and  $C_2$ .

Therefore  $\beta_1$  and  $\alpha_3$  are minimized. Similarly if the solution is to be between  $D_1$  and  $D_2$ , then  $\beta_2$  and  $\alpha_4$  are minimized. By assigning proper weights, we obtain the objective function (3.18).

### 3.9 The Algorithm for Interval Goal Network Flow Problem:

The algorithm for interval goal network flow problem is similar to that of weighted goal network flow problem except for a few modifications.

The procedure to trace the path to reach an optimum solution is same as that of WGLF algorithm but the paths that these two algorithms take to obtain the optimum solutions are different. In interval goal problem, the decision maker specifies a range of aspiration levels for each of the goals and weighting factors for deviations from the ranges. The ranges specified form a rectangular region  $R$  in  $E$  whose four corner points are  $(C_1, D_1)$ ,  $(C_2, D_1)$ ,  $(C_2, D_2)$ ,  $(C_1, D_2)$  as shown in Fig. 3.7. The algorithm obtains an optimum solution  $(X, \alpha_3, \alpha_4, \beta_1, \beta_2)$  by minimizing the sum of weighted deviations.

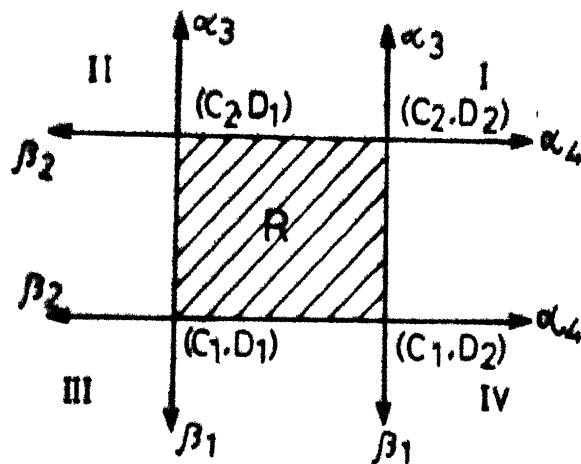


Fig. 3.7 Aspiration levels and deviations for IGNF problem

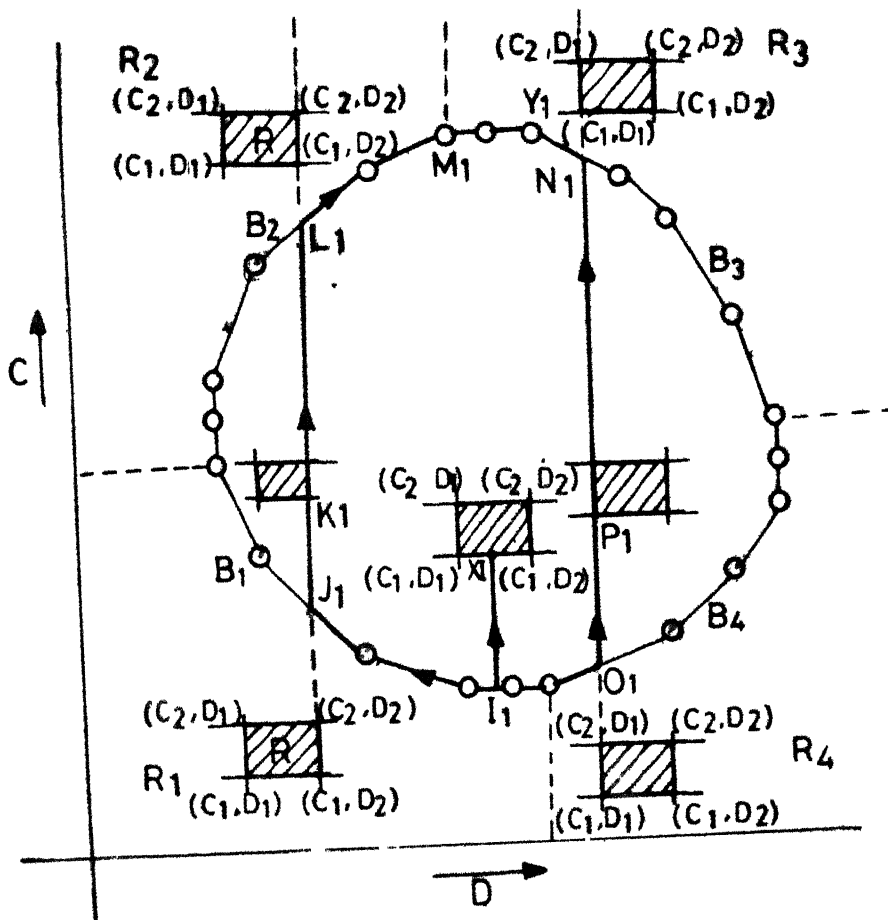


Fig 3.8 Various paths traced by IGNF algorithm to obtain optimum solutions

From above, it can be inferred that for different positions of R, the IGNF algorithm considers one of the corner co-ordinates of R as aspiration levels and obtains an optimum solution like the WGNF algorithm. Therefore, all the theorems proved for WGNF algorithm are also valid for IGNF algorithm.

### 3.10 Numerical Example:

In this section a small network flow problem is solved to illustrate the various steps of IGNF algorithm. The network is shown in Fig. 3.4. The aspiration level ranges and the weighting factors specified for this problem are as follows:  
 $C_1 = 91$ ,  $C_2 = 96$ ,  $D_1 = 80$ ,  $D_2 = 90$ ,  $w_3 = 0.2$ ,  $w_4 = 0.3$ ,  
 $r_1 = 0.15$ ,  $r_2 = 0.35$ .

The steps of the algorithm are summarized in Table 3.2. The  $\uparrow$  indicates the basic pivot arc and  $\downarrow$  indicates the basic arc leaving the basis. The algorithm traces I,P,Q path as shown in Fig. 3.5 and obtains  $C = 91$  and  $D = 90$  as an optimum solution. Q represents an optimum solution, in Fig. 3.5.

### 3.11 Computational Results:

The algorithms proposed in this chapter were coded, debugged and tested in Fortran - 10 on DEC-1090 multi-programming, time sharing computer system. A number of randomly generated network problems were solved with different number



Table 3.2: Solution of the IGNF Problem.

It.	C	D	Z	Basic Tree Arcs						Non-tree Arcs					
				(1,j)	(1,2)	(1,3)	(2,4)	(4,6)	(5,6)	(2,5)	(2,3)	(3,4)	(3,5)	(5,4)	
1	63	94	5	$x_{ij}$	5	0	2	3	2	3	0	0	0	0	
				$\bar{c}_{ij}$	0	0	0	0	0	4	3	7	2	9	
				$\bar{d}_{ij}$	0	0	0	0	0	-3	10	-4	<del>-3</del> $\uparrow(p,q)$	7	
				(i,j)	(1,2)	(1,3)	(2,4)	(5,6)	(4,6)	(2,5)	(2,3)	(3,4)	(3,5)	(5,4)	
2	0.67	65.7	90	$x_{ij}$	3.67	1.33	0.67	4.33	0.67	3	0	0	4.33	0	
				$\bar{c}_{ij}$	0	0	0	0	0	4	1	-4	2	9	
				$\bar{d}_{ij}$	0	0	0	0	0	-3	$\uparrow(x,y)$	3	-3 $\uparrow(p,q)$	-7	
				(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)	(5,4)	
3	67.6	90	3.51	$x_{ij}$	3.2	1.8	0	2	5	3	0	0	0	0	
				$\bar{c}_{ij}$	0	0	0	0	0	6	1	7	-2	11	
				$\bar{d}_{ij}$	0	0	0	0	0	$\downarrow$ $\uparrow(x,y)$	-6 $\uparrow(x,y)$	10 $\uparrow(p,q)$	-4	-2	11
				(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)	(5,4)	
4	91	90	0	$x_{ij}$	2	3	0	5	5	0	2	0	0	0	
				$\bar{c}_{ij}$	0	0	0	0	0	-6	1	7	-2	11	
				$\bar{d}_{ij}$	0	0	0	0	0	6	10 $\uparrow(p,q)$	-4	3	4	

of nodes and arcs and computational times were noted. A network generator was used for computational study which generates well-structured networks. The program first generates a skeleton network for a specified width and length and then adds arcs randomly until the network contains a specified number of arcs.

Data structures based on augmented threaded index method[8] was implemented for storing the spanning tree. The tree was stored by means of thread indices, predecessor indices and the number of successors, each requiring an array of size  $n$ . Thread of a node  $i$  is a node that will be scanned in depth first search order of the sub-tree rooted at  $i$ . Predecessor of node  $i$  is the first node on the path from  $i$  to the root node of the tree. The number of successors of node  $i$  is the number of nodes in the subtree rooted at  $i$ . Link of a node  $i$  is the first arc on the path from  $i$  to the root node of the tree. If the arc is incident to  $i$ , it is stored as positive. If the arc is incident from  $i$ , it is stored as negative.

When an arc leaves the basis, it results in the formation of a hanging sub-tree. Thread indices help in scanning all the nodes of this sub-tree and thereby updating the dual variables. Predecessor indices are used for determining the minimum flow to be augmented, the leaving arc, and for updating the flows in the cycle formed, when a non-basic arc is added

to the basis. The number of successors for each node is maintained. It helps in determining the common node for the two paths when traversed from two ends of the entering arc towards the root node. Once the common node is known, it is easy to determine the minimum flow to be augmented in the cycle, leaving arc and updating the flow as well as the basis structure. Link is an array of size  $n$  that stores the arcs belonging to the basic tree. The storage for preserving the flows in arcs was reduced from  $m$  to  $n$ . This was achieved by storing the flows of basic arcs requiring an array of size  $n$ . The remaining non-basic arcs exist either at their lower or upper bounds, respectively. If the non-basic arc is at its upper bound, its capacity is made negative otherwise its capacity remains positive.

The main emphasis with computational results was laid on (i) to check the number of iterations performed by the algorithm and (ii) the computational time taken by the algorithms to get the optimum solutions. Problems sizes ranging from 10 nodes and 40 arcs to 200 nodes and 2500 arcs which includes both the sparse as well as dense networks were considered for computational study. Arc capacities were randomly generated between 5 to 100 whereas  $c_{ij}$ 's and  $d_{ij}$ 's were randomly generated between 1 to 50. Each problem was solved for three different aspiration levels and the computational times as well as the number of iterations are noted in Tables 3.3 and

3.4. About 20 problems of different sizes were solved for each of the algorithms. By comparing the columns (6), (8) and (10) of Table 3.3, we can infer that the time taken for solving  $(C_1, D_1) \in R_2 \cup R_3$  is more than that for solving  $(C_1, D_1) \in R_1 \cup R_4$  or  $(C_1, D_1) \in S$ . This is because, the optimum solution for  $(C_1, D_1) \in R_2 \cup R_3$  lies on  $B_2 \cup B_3$  and thereby traversing a longer path. No comparison can be made between (6) and (8) of Table 3.3 as the time taken to solve  $(C_1, D_1) \in R_1 \cup R_4$  and  $(C_1, D_1) \in S$  depends on the values of  $C_1$  and  $D_1$ . One can infer similar results by comparing columns (6), (8) and (10) of Table 3.4 of IGNF problem. The reason being that WGNF algorithm and IGNF algorithm behave in a similar manner.

It is evident from the tables that both the algorithms can solve quite large problems in reasonable amount of time. For instance, problem of size 400 nodes and 2500 arcs was solved in about  $2\frac{1}{2}$  minutes by both the algorithms. Infact the time taken by both the algorithms are comparable because IGNF algorithm is similar to WGNF algorithm except for a few modifications which will not alter computational times significantly.

Both the algorithms suggested by us are able to solve practically large problems in a reasonable amount of computer time. From the literature review we can conclude that no work was done in the field of goal programming techniques applied to

Table: 3.3: Computational Times of WGNF Algorithm.  
(Execution Times in Seconds on DEC-1090 System)

Width	Length	Nodes	Arcs	$(C_1, D_1) \epsilon R_4$		$(C_1, D_1) \epsilon S$		$(C_1, D_1) \epsilon R_3$	
				ITER	TIME	ITER	TIME	ITER	TIME
3	3	10	40	36	0.07	34	0.053		
3	5	15	70	82	0.17	102	0.14	58	0.1
5	5	25	100	106	0.27	151	0.23	82	0.18
5	5	25	200	84	0.34	71	0.24	192	0.42
5	10	50	200	282	1.45	369	1.56	177	0.82
5	10	50	400	141	1.17	100	0.65	379	1.92
5	15	75	300	435	3.22	386	1.24	288	2.58
5	15	75	600	230	2.74	277	3.21	535	3.98
10	10	100	400	580	5.73	1115	4.51	553	7.55
10	10	100	1000	313	6.1	303	5.35	778	7.53
								383	7.59

Table 3.3 continued

Width	Length	Nodes	Arcs	$(C_1, D_1) \varepsilon_3 UR_4$		$(C_1, D_1) \varepsilon_5$		$(C_1, D_1) \varepsilon_8 UR_3$	
				ITER	TIME	ITER	TIME	ITER	TIME
5	25	75	500	785	9.56	332	3.74	947	11.38
5	25	75	1200	1652	48.92	13370	99.64	3246	93.44
10	15	150	600	1062	15.42	1262	11.96	1369	19.59
10	15	150	1500	3226	118.45	4172	115.15	4334	155.5
7	25	175	700	1315	22.19	1711	23.28	1701	28.33
7	25	175	1600	3042	133.40	9480	147.19	4532	173.83
10	20	200	1000	403	8.02	672	8.93	915	20.40
10	20	200	2000	3674	146.39	9521	153.26	4756	181.57
20	20	400	1500	3709	139.37	9377	146.26	4375	173.69
20	20	400	2500	5213	172.56	11361	199.52	6857	208.18

Table 3.4: Computational Times of IGNF Algorithm.  
(Execution Times in Seconds on DEC-1090 Systems)

Width	Length	Nodes	Arcs	$R \in P_1$		$R \in S$		$R \in R_2$	
				ITER	TIME	ITER	TIME	ITER	TIME
3	3	10	40	36	0.07	34	0.053		
3	5	15	70	82	0.17	102	0.139	58	0.11
5	5	25	100	114	0.32	124	0.22	82	0.18
5	5	25	200	88	0.39	136	0.36	167	0.47
5	10	50	200	243	1.26	541	1.86	191	0.95
5	10	50	400	158	1.37	610	1.91	391	2.03
5	15	75	300	397	2.93	302	1.14	379	2.71
5	15	75	600	237	2.89	280	3.07	535	3.98
10	10	100	400	607	6.09	410	2.16	602	8.40
						791		791	7.78

Table 3.4 continued

Width	Length	Nodes	Arcs	$R \in R_1$		$R \in S$		$R \in R_2$	
				ITER	TIME	ITER	TIME	ITER	TIME
10	10	100	1000	313	6.06	458	5.52	1139	12.33
5	25	125	500	387	4.57	384	3.95		
5	25	125	1200	1603	47.89	11053	85.37	915	11.13
10	15	150	600	1096	16.05	2338	12.41	3085	89.20
10	15	150	1500	3361	123.31	11944	125.61	1374	19.72
7	25	175	700	1370	23.27	2695	28.91	4444	159.60
7	25	175	1600	3535	138.60	13692	186.14	1739	29.17
10	20	200	1000	426	8.648	1437	15.20	4633	177.83
10	20	200	2000	3572	141.35	9407	150.82	3507	26.31
20	20	400	1500	3657	138.07	9938	167.55	4947	183.58
20	20	400	2500	5008	165.09	10021	195.83	4489	179.56
								6517	203.18



network flow problems. This made us to explore this field, and suggest exact algorithms for weighted goal and interval goal network flow problems. One disadvantage with goal programming techniques is, the decision maker should specify proper weighting factors otherwise he may not obtain the preferred solution.

## REFERENCES

1. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained minimum cost flow problem, Research paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).
2. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained maximum flow problem, working paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).
3. Ambrose, G., Hansen, D.R., and Lucien, D., Multi-objective Decision Analysis with Engineering and Business Applications, John Wiley and Sons, (1982).
4. Chen, S., and Saigal, R., A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints, Networks 7 (1977), 59-80.
5. Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J. (1963).
6. Glover, F., Network Applications in Industry and Government.
7. Ignizio, Jame, P., An Approach to the Modelling and Analysis of Multi-objective Generalised Networks, EJOR (Netherlands) 12 (1933) 4, 357-361.
8. Kennington, J.L., and Helgason, R.V., Algorithms for Networks Programming, John Wiley and Sons, Inc., New York, (1980).
9. Klingman, D., and Hultz, J., Solving Constrained Generalized Network Problems, Research Report CCS 257 Centre for Cybernetics Studies, University of Texas, Austin, Texas, Nov. 1976.
10. Klingman, D., and Mote, J., Solution Approaches for Network Flow Problems with Multiple Criteria, AMS (India) 1 (1932), Jan.
11. Lawler, E., Combinatorial Optimization, Networks and Matroids, Holt, Rinehart and Winston (1976).
12. Takashi, K., The Lexico-shortest Route Algorithm for Solving the Minimum Cost Flow Problem with an Additional Linear Constraint, Journal of Operations Research Society of Japan, Vol. 26, No. 3, Sept. 1983.
13. Zeleny, M., Multi-criteria Decision Making, McGraw-Hill Book Company. 1982.

```

*****
PROGRAM TO SOLVE BI-CRITERIA MINIMUM COST FLOW PROBLEM BY
GOAL PROGRAMMING
*****
INTEGER WIDTH, SEED, ROOT, SOURCE, SINK, PREV, COUNT, FRONT, START,
1 FINISH, ENTER, V, STATUS, O, OP, Z, X, R,
2 TIME1, TIME2, TIME, CMIN, TR,
3 PRED(405), THREAD(405), PI(405), POINT(405), PID(0:405),
4 SCAN(405), HEAD(3000), CAP(3000), CDST(3000), BUD(0:3000),
5 PO, RS, TAILPO, HEADPO, TAILRS, HEADRS, CODE, CODE1, CODE2, OVER
REAL MAX, MAX1, MAX2, MUE, MUEMAX, ICAP, IFLOW, MINIM, JFLOW
DIMENSION NUMBER(405), LINK(405), LIST(405), FLOW(405), SUM(405)
COMMON ITAIL, IHEAD, STATUS, DELTA, ICAP, LEG, ENTER, CMIN, DMIN, IW,
1 LINK, FLOW, PRED, SUM, CAP, NUMBER, THREAD, PI, PID, IFLOW, K
2 LEAVE, RFLOW
TYPE 10
10 FORMAT(1X, '>>>', '$')
ACCEPT *, WIDTH, LENGTH, M, SEED, V, C1, D1
CALL SETRAN(SEED)
CALL NG4(WIDTH, LENGTH, N, M, SOURCE, SINK, POINT, HEAD)
DO 20 J=1, M
CAP(J)=IRAN(5, 100)
COST(J)=IRAN(1, 50)
20 BUD(J)=IRAN(1, 50)
READ(21, *) W1, W2, W3, W4
*****
* THIS PART OF PROGRAM FORMS THE BASIS TREE AND CALCULATES THE *
* DUAL VARIABLES. *
*****
CALL RTIME(TIME1)
OVER=0
S=1.0E-5
FLAG=0
RFLAG=0
IFLOW=0
DO 30 I=1, N
PRED(I)=0
PI(I)=0
PID(I)=0
30 CONTINUE
LARGE=10000000
C=0
MINIM=1.0E+14
BUD(0)=0
ROOT=SOURCE
MA=M+1
CAP(MA)=LARGE
CDST(MA)=LARGE
BUD(MA)=0.0
PRED(SINK)=SOURCE
LINK(SINK)=MA
PREV=SINK; TOP=1; COUNT=0
KOUNT=1
40 SCAN(TOP)=ROOT
IF(TOP.EQ.0) GO TO 70
I=SCAN(TOP)
IF(I.GT.0) GO TO 50
I=-I
NUMBER(I)=COUNT-LIST(I)
TOP=TOP-1
GO TO 40
50 LIST(I)=COUNT
COUNT=COUNT+1
THREAD(PREV)=I
PREV=I
SCAN(TOP)=-I
IF(KOUNT.GE.N) GO TO 40
IX=POINT(I)
IY=POINT(I+1)-1
IF(IX.GT.IY) GO TO 40
DO 60 J=IX, IY
K=HEAD(J)
IF(PRED(K).NE.0) GO TO 60
PRED(K)=I
LINK(K)=J
TOP=TOP+1
SCAN(TOP)=K
KOUNT=KOUNT+1
60 CONTINUE
GO TO 40
70 THREAD(PREV)=SINK
THREAD(SINK)=SOURCE
NUMBER(SINK)=1
NUMBER(SOURCE)=N
80 CONTINUE
PI(1)=0
PID(1)=0
PID(0)=0
PI(SINK)=LARGE

```

```

00917
P10(SINK)=C
FLOW(1)=0
FLOW(SINK)=V
C=V+LARGE
D=0.0
I=SINK
90 K=THREAD(I)
PI(K)=PI(PRED(K))+COST(LINK(K))
PID(K)=PID(PRED(K))+BUD(LINK(K))
FLOW(K)=0
IF(K.EQ.PREV)GO TO 100
I=K
GO TO 90
100 CONTINUE
CODE=5
110 RSIGN=-1
CMINIM=C
DMINIM=0
*****
* THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON - BASIC *
* ARC PO FOR FLOW AUGMENTATION. *
*****
120 SIGN=-1
MUEMAX=LARGE
IF(D1.LT.DMINIM)SIGN=1
130 FLOWPO=0
ITER=ITER+1
IF(D1.EQ.0.AND.C.GE.C1)GO TO 920
IF(C1.EQ.C.AND.0.NE.D1.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 920
Q=ABS(D-D1)
IF((CODE.LE.2).AND.(Q.LE.5))GO TO 500
ENTER=0
DO 230 I=1,N
START=POINT(I)
FINISH=POINT(I+1)-1
IF(START.GT.FINISH)GO TO 230
DO 230 J=START,FINISH
K=HEAD(J)
IF(CAP(J).LT.0)GO TO 140
CBAR=COST(J)+PI(I)-PI(K)
DBAR=BUD(J)+PID(I)-PID(K)
STATUS=1
GO TO 150
140 CBAR=PI(K)-PI(I)-COST(J)
DBAR=PID(K)-PID(I)-BUD(J)
STATUS=-1
150 GO TO(160,210,170,210,200)CODE
160 IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
GO TO 180
170 IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
180 ENTER=J
CMIN=CBAR
DMIN=DBAR
STATPO=STATUS
IHEAD=K
ITAIL=I
PO=J
190 GO TO 270
200 IF(CBAR.GE.0.OR.MUEMAX.LE.CBAR)GO TO 230
MUEMAX=CBAR
GO TO 220
210 IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
MUE=(RSIGN*CBAR)/(SIGN*DBAR)
IF(MUE.GE.MUEMAX)GO TO 230
MUEMAX=MUE
220 ENTER=J
CMIN=CBAR
DMIN=DBAR
STATPO=STATUS
PO=J
IHEAD=K
ITAIL=I
230 CONTINUE
IF(CODE.EQ.5.AND.ENTER.NE.0)GO TO 270
IF(CODE.EQ.5.AND.ENTER.EQ.0)CODE=1
IF(ENTER.NE.0)GO TO 260
IF(CODE.EQ.4.AND.ENTER.EQ.0)GO TO 920
CODE=CODE+1
IF(CODE.NE.4)GO TO 120
240 SIGN=-SIGN
RSIGN=-RSIGN
MUEMAX=0
GO TO 130
260 MAX1=(C1-C)/CMIN
MAX2=(D1-D)/DMIN
MAX=MAX1
IF((MAX1.GT.MAX2.AND.MAX2.GT.0).OR.MAX1.LE.0)MAX=MAX2

```

```

*****
* THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE
* AMOUNT OF FLOW TO BE AUGMENTED.
*****

```

```

270 ICAP=CAP(ENTER)
    LEG=0
    FLOWPO=IABS(CAP(PO))
    IF(STATPO.EQ.1)FLOWPO=0
    IF(FLAG.EQ.0)GO TO 280
    FLOWPO=TFLOW
    FLAG=0
280 IF(ICAP.LT.0)GO TO 290
    STATUS=1
    IX=ITAIL
    IY=IHEAD
    DELTA1=ICAP-FLOWPO
    GO TO 300
290 ICAP=-ICAP
    CMIN=-CMIN
    DMIN=-DMIN
    STATUS=-1
    IX=IHEAD
    IY=ITAIL
    DELTA1=FLOWPO
    DELTA=DELTA1
    RFLOW=FLOWPO
310 IF(IX.EQ.IY)GO TO 390
    IF(NUMBER(IX).GT.NUMBER(IY))GO TO 350
    J=LINK(IX)
    IF(J.LT.0)GO TO 320
    CHANGE=CAP(J)-FLOW(IX)
    GO TO 330
320 CHANGE=FLOW(IX)
330 IF(CHANGE.GE.DELTA)GO TO 340
    DELTA=CHANGE
    K=IX
    LEG=1
340 IX=PRED(IX)
    GO TO 310
350 J=LINK(IY)
    IF(J.LT.0)GO TO 360
    CHANGE=FLOW(IY)
    GO TO 370
360 CHANGE=CAP(-J)-FLOW(IY)
370 IF(CHANGE.GE.DELTA)GO TO 380
    DELTA=CHANGE
    K=IY
    LEG=-1
380 IY=PRED(IY)
    GO TO 310
390 CONTINUE
    IW=IX
    DELTA1=DELTA
    IF(CODE.NE.5)GO TO 410
    STATUS=STATPO
    ENTER=PO
    C=C+CMIN*STATUS*DELTA
    D=D+DMIN*STATUS*DELTA
    IDIR=-1
    IF(D1.GT.D)IDIR=1
    IF(DELTA1.LT.ICAP)GO TO 400
    CALL AUGMEN
    GO TO 110
400 CALL UPDATE
    GO TO 110
410 IF(DELTA1.LE.MAX)GO TO 460
    DELTA=MAX
    ADD=CMIN*STATUS*DELTA
    ADD1=DMIN*STATUS*DELTA
    C=C+ADD
    D=D+ADD1
    CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
    C=C-ADD
    D=D-ADD1
    IF(OVER.EQ.1)GO TO 920
    C=C+ADD
    D=D+ADD1
    IF(ABS(D-D1).LE.S.AND.CODE.GE.3)GO TO 430
    IF(D.NE.D1.AND.CODE.LE.2)GO TO 430
    IF(ABS(C-C1).GT.S.AND.CODE.GE.3)GO TO 430
420 CALL AUGMEN
    CAP(ENTER)=-CAP(ENTER)
    GO TO 130
430 DELTA=MAX1
    IF(MAX.EQ.MAX1)DELTA=MAX2
    IF(DELTA.LT.0.OR.DELTA.GE.DELTA1)GO TO 450
    DELTA=DELTA-MAX
    ADD1=CMIN*STATUS*DELTA

```



```

00915
02650 ADDL=DMIN*STATUS*DELTA
02660 C=C+ADD1
02670 D=D+ADD2
02710 IF(ABS(C-C1).LE.S.AND.CODE.GE.3)GO TO 420
02720 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
02730 IF(OVER.EQ.1)GO TO 440
02740 DELTA=DELTA+MAX
02750 CALL AUGMEN
02760 GO TO 920
02770 440 DELTA=MAX
02780 C=C-ADD1
02790 D=D-ADD2
02800 CALL AUGMEN
02810 GO TO 920
02820 450 DELTA=DELTA1-MAX
02830 460 ADD=CMIN*STATUS*DELTA
02840 ADD1=DMIN*STATUS*DELTA
02850 C=C+ADD
02860 D=D+ADD1
02870 DELTA=DELTA1
02880 IF(DELTA1.LE.S)GO TO 480
02890 470 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
02900 C=C-ADD
02910 D=D-ADD1
02920 IF(OVER.EQ.1.AND.(DELTA.EQ.(DELTA1-MAX)))GO TO 440
02930 IF(OVER.EQ.1)GO TO 920
02940 C=C+ADD
02950 D=D+ADD1
02960 480 IF(LEG.EQ.0)CALL AUGMEN
02970 IF(LEG.NE.0)CALL UPDATE
02980 490 IF(CODE.EQ.4)GO TO 250
02990 GO TO 120
03000 *****
03010 * THIS PART OF THE PROGRAM SELECTS THE NON - BASIC ARCS FOR *
03020 * SIMULTANEOUS FLOW AUGMENTATION IN TWO CYCLES. *
03030 *****
03040 500 CBPQ=CMIN
03050 DBPQ=DMIN
03060 TAILPQ=ITAIL
03070 HEADPQ=IHEAD
03080 FLOWPQ=DELTA
03090 IF(STATPQ.EQ.-1)FLOWPQ=-CAP(PQ)-DELTA
03100 CODE2=1
03110 510 RS=0
03120 ITER=ITER+1
03130 DO 530 I=1,N
03140 530 SUM(I)=0.0
03150 IF(ABS(C-C1).LE.S)GO TO 920
03160 DO 560 J=1,N
03170 START=POINT(I)
03180 FINISH=POINT(I+1)-1
03190 IF(START.GT.FINISH)GO TO 560
03200 DO 560 J=START,FINISH
03210 K=HEAD(J)
03220 IF(CAP(J).LT.0)GO TO 540
03230 CBAR=COST(J)+PI(I)-PI(K)
03240 DBAR=BUD(J)+PID(I)-PID(K)
03250 STATUS=1
03260 GO TO 550
03270 540 CBAR=PI(K)-PI(I)-COST(J)
03280 DBAR=PID(K)-PID(I)-BUD(J)
03290 STATUS=-1
03300 550 IF(DBAR.EQ.0)GO TO 560
03310 IF(DBAR/DBPQ.GE.0)GO TO 560
03320 IF((CBPQ-(DBPQ*CBAR)/DBAR).LE.0)GO TO 560
03330 RS=J
03340 TAILRS=I
03350 HEADRS=K
03360 STATRS=STATUS
03370 RATIO=-DBPQ/DBAR
03380 CBR=CBAR
03390 DBR=DBAR
03400 FLOWRS=0.0
03410 IF(STATRS.EQ.-1)FLOWRS=-CAP(RS)
03420 GO TO 590
03430 560 CONTINUE
03440 IF(CODE2.EQ.2.AND.RS.EQ.0)GO TO 570
03450 CBPQ=-CBPQ
03460 DBPQ=-DBPQ
03470 CAP(PQ)=-CAP(PQ)
03480 STATPQ=-STATPQ
03490 CODE2=CODE2+1
03500 GO TO 520
03510 570 CODE=4
03520 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
03530 IF(CBPQ.GT.0)GO TO 580
03540 CBPQ=-CBPQ
03550 DBPQ=-DBPQ
03560 CAP(PQ)=-CAP(PQ)

```

```

00910
580 IF(CBPQ.LT.0.AND.IDIR.EQ.-1)RFLAG=-1
IF(CBPQ.GT.0.AND.IDIR.EQ.1)RFLAG=+1
IF(RFLAG.NE.0)SIGN=-SIGN
FLAG=1
IFLOW=FLOWPQ
GO TO 240
*****
* THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE *
* MINIMUM FLOW TO BE AUGMENTED IN TWO CYCLES SIMULTANEOUSLY *
*****
590 CODE1=1
DELTA1=FLOWPQ
IF(STATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
LEG=0
DELTA2=IABS(CAP(RS))/RATIO
IF(DELTA2.LT.DELTA1)DELTA1=DELTA2
600 ENTER=PQ
STATUS=STATPQ
RIN=1
INC=1
ITAIL=TAILPQ
IHEAD=HEADPQ
610 IF(STATUS.EQ.-1)GO TO 620
IX=ITAIL
IY=IHEAD
GO TO 630
620 IX=IHEAD
IY=ITAIL
630 IF(IX.EQ.IY)GO TO 790
IF(NUMBER(IX).GT.NUMBER(IY))GO TO 710
IF(CODE1.EQ.3)GO TO 660
J=LINK(IX)
IF(J.LT.0)GO TO 640
SUM(IX)=SUM(IX)+RIN
GO TO 650
640 SUM(IX)=SUM(IX)-RIN
650 IF(CODE1.EQ.1)GO TO 700
660 IF(SUM(IX))670,700,680
670 DELTA2=-FLOW(IX)/SUM(IX)
GO TO 690
680 J=IABS(LINK(IX))
DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
690 IF(DELTA2.GE.DELTA1)GO TO 700
DELTA1=DELTA2
LEG=INC
K1=IX
TOTAL=FLOW(K1)+DELTA1*SUM(K1)
700 IX=PREQ(IX)
GO TO 630
710 IF(CODE1.GT.2)GO TO 740
J=LINK(IY)
IF(J.LT.0)GO TO 720
SUM(IY)=SUM(IY)-RIN
GO TO 730
720 SUM(IY)=SUM(IY)+RIN
730 IF(CODE1.EQ.1)GO TO 780
740 IF(SUM(IY))750,780,760
750 DELTA2=-FLOW(IY)/SUM(IY)
GO TO 770
760 J=IABS(LINK(IY))
DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
770 IF(DELTA2.GE.DELTA1)GO TO 780
DELTA1=DELTA2
LEG=-INC
K1=IY
TOTAL=FLOW(K1)+DELTA1*SUM(K1)
780 IY=PREQ(IY)
GO TO 630
790 IF(CODE1.EQ.1)IN1=IX
IF(CODE1.EQ.2)IN2=IX
CODE1=CODE1+1
IF(CODE1.GT.3)GO TO 800
IF(CODE1.EQ.3)GO TO 600
ENTER=RS
ITAIL=TAILRS
IHEAD=HEADRS
STATUS=STATRS
RIN=RATIO
INC=2
GO TO 610
*****
* THIS PART OF THE PROGRAM UPDATES THE FLOW IN BOTH THE CYCLES *
* BY CALLING AN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES *
* THE ARC PQ. *****
800 DELTA2=(C1-C)/(CBPQ+RATIO*CBRS)
IF(DELTA2.GT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
C=C+DELTA1*CBPQ+RATIO*DELTA1*CBRS
D=D+DELTA1*DBPQ+RATIO*DELTA1*DBRS

```

```

00910
4470      PI(SIZE)=GARGE
4471      IF (SIZE.GE.0)GO TO 840
4472      DELTA=DELTA1*RATIO
4473      ITAIL=TAILRS
4474      IHEAD=HEADRS
4475      STATUS=STATRS
4476      FLOWRS=FLOWRS+STATRS*DELTA
4477      IW=IW2
4478      ENTER=RS
4479      CALL AUGMEN
4480      DELTA=DELTA1
4481      ITAIL=TAILPO
4482      IHEAD=HEADPO
4483      STATUS=STATPO
4484      FLOWPO=FLOWPO+STATPO*DELTA
4485      IW=IW1
4486      ENTER=PO
4487      CALL AUGMEN
4488      IF (FLOWRS.NE.0.AND.FLOWRS.NE.IABS(CAP(RS)))GO TO 820
4489      IF (ABS(FLOWRS).GE.S.AND.ABS(FLOWRS-IABS(CAP(RS))))GO TO 830
4490      STATPO=-STATPO
4491      CBPO=-CBPO
4492      DBPO=-DBPO
4493      IF (FLOWPO.LE.S.OR.ABS(FLOWPO-IABS(CAP(PO))).LE.S)GO TO 510
4494      STATPO=-STATPO
4495      CAP(PO)=-CAP(PO)
4496      CBPO=-CBPO
4497      DBPO=-DBPO
4498      GO TO 510
4499      810 FLOWPO=FLOWRS
4500      CBPO=-CBRS
4501      DBPO=-DBRS
4502      PO=RS
4503      TAILPO=TAILRS
4504      HEADPO=HEADRS
4505      STATPO=-STATRS
4506      830 IF (FLOWPO.LE.0.OR.ABS(FLOWPO-IABS(CAP(PO))).LE.S)GO TO 510
4507      CAP(PO)=-CAP(PO)
4508      STATPO=-STATPO
4509      CBPO=-CBPO
4510      DBPO=-DBPO
4511      GO TO 510
4512      840 IF (IABS(LEG).EQ.2)GO TO 880
4513      ITAIL=TAILRS
4514      IHEAD=HEADRS
4515      STATUS=STATRS
4516      DELTA=DELTA1*RATIO
4517      FLOWRS=FLOWRS+STATRS*DELTA
4518      IW=IW2
4519      ENTER=RS
4520      CALL AUGMEN
4521      ITAIL=TAILPO
4522      IHEAD=HEADPO
4523      STATUS=STATPO
4524      DELTA=DELTA1
4525      IW=IW1
4526      ENTER=PO
4527      CMIN=CBPO
4528      DMIN=DBPO
4529      IF (STATUS.EQ.1)GO TO 850
4530      CMIN=-CMIN
4531      DMIN=-DMIN
4532      850 ICAP=IABS(CAP(PO))
4533      K=K1
4534      RFLOW=FLOWPO
4535      CALL UPDATE
4536      CAP(LEAVE)=-IABS(CAP(LEAVE))
4537      IF (TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
4538      FLOWPO=FLOWRS
4539      PO=RS
4540      TAILPO=TAILRS
4541      HEADPO=HEADRS
4542      CBPO=COST(PO)+PI(TAILPO)-PI(HEADPO)
4543      DBPO=BUD(PO)+PID(TAILPO)-PID(HEADPO)
4544      IF (STATRS.EQ.1)GO TO 860
4545      CBPO=-CBPO
4546      DBPO=-DBPO
4547      860 STATPO=-STATRS
4548      CBPO=-CBPO
4549      DBPO=-DBPO
4550      GO TO 830
4551      870 ENTER=PO
4552      880 ITAIL=TAILPO
4553      IHEAD=HEADPO
4554      STATUS=STATPO
4555      DELTA=DELTA1
4556      IW=IW1
4557      FLOWPO=FLOWPO+STATPO*DELTA1
4558      CALL AUGMEN
4559      ITAIL=TAILRS

```



```

0091A
PI(SINK)=LARGE
HEAD=HEADRS
STATUS=STATRS
DELTA=DELTA1*RATIO
IW=IN2
LEG=LEG/2
ENTER=RS
K=K1
CMIN=CBRS
DMIN=DBRS
IF(STATUS.EQ.1)GO TO 890
CMIN=-CMIN
DMIN=-DMIN
890 ICAP=IABS(CAP(RS))
REFLOW=FLOWRS
CALL UPDATE
CAP(LEAVE)=-IABS(CAP(LEAVE))
IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
IF(STATPQ.EQ.1)GO TO 900
CBPQ=-CBPQ
DBPQ=-DBPQ
900 STATPQ=-STATPQ
CBPQ=-CBPQ
DBPQ=-DBPQ
910 GO TO 810
920 CALL RTIME(TIME2)
WRITE(22,930)C,D,TIME,ITER
930 FORMAT(1X,'COST=',F14.4,' BUDGET=',F14.4,' TIME=',I8,' ITER=',I8)
STOP
END

```

```

0001 *****
0002 * THIS SUBROUTINE UPDATES THE OBJECTIVE FUNCTION OF FIXED *
0003 * WEIGHTED GOAL PROGRAMMING. *
0004 *****
0005 SUBROUTINE UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,
0006 1 RFLAG)
0007 REAL MIN,MINIM
0008 INTEGER OVER
0009 IF(DELTA.LE.(1.0E-6))GO TO 80
0010 IF(RFLAG)30,5,10
0011 5 IF(IDIR)10,80,30
0012 10 IF(C.GT.C1)GO TO 40
0013 MIN=N3*(C1-C)+W2*(D-D1)
0014 70 OVER=1
0015 IF(MIN.GE.MINIM)GO TO 80
0016 MINIM=MIN
0017 OVER=0
0018 80 RETURN
0019 40 MIN=N1*(C-C1)+W2*(D-D1)
0020 GO TO 70
0021 30 IF(C.GT.C1)GO TO 50
0022 MIN=N3*(C1-C)+W4*(D1-D)
0023 GO TO 70
0024 50 MIN=N1*(C-C1)+W4*(D1-D)
0025 GO TO 70
0026 END

```

```

0001 *****
0002 * THIS PROGRAM OBTAINS AN OPTIMUM SOLUTION FOR INTERVAL GOAL *
0003 * PROGRAMMING APPLIED TO BI-CRITERIA NETWORK FLOW PROBLEM. *
0004 *****
0005 INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
0006 1 FINISH,ENTER,V,STATUS,0,OP,2,X,R,
0007 2 TIME1,TIME2,TIME,CMIN,TR
0008 3 PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
0009 4 SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
0010 5 P2,RS,TAILP2,HEADP2,TAILRS,HEADRS,CODE,CODE1,CODE2,OVER
0011 REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,MINIM,JFLOW
0012 DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
0013 COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
0014 1 LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K,
0015 2 LEAVE,RFLOW
0016 TYPE 10
0017 10 FORMAT(1X,'>>>')
0018 ACCEPT *,WIDTH,LENGTH,M,SEED,V,C1,D1,C2,D2
0019 CALL SETRAN(SEED)
0020 CALL NG4(WIDTH,LENGTH,N,M,SOURCE,SINK,POINT,HEAD)
0021 DO 20 J=1,M
0022 CAP(J)=IRAN(5,100)
0023 COST(J)=IRAN(1,50)
0024 20 BUD(J)=IRAN(1,50)
0025 READ(21,*)W1,W2,W3,W4
0026 CALL RTIME(TIME1)
0027 *****
0028 * THIS PART OF THE PROGRAM FORMS THE BASIS TREE ALONG WITH THE
0029 * PREDECESSOR,THREAD INDICES, NO. OF SUCCESSORS AND CALCULATES
0030 * THE DUAL VARIABLES PI'S AND PID'S.
0031 *****
0032 OVER=0
0033 FLAG=0
0034 S=1.0E-5
0035 RFLAG=0
0036 TFLOW=0
0037 DO 30 I=1,N
0038 PRED(I)=0
0039 PI(I)=0
0040 PID(I)=0
0041 30 CONTINUE
0042 LARGE=1000000
0043 C=0
0044 MINIM=1.0E+14
0045 BUD(0)=0
0046 ROOT=SOURCE
0047 MA=M+1
0048 CAP(MA)=LARGE
0049 COST(MA)=LARGE
0050 BUD(MA)=0.0
0051 PRED(SINK)=SOURCE
0052 LINK(SINK)=MA
0053 PREV=SINK;TOP=1;COUNT=0
0054 KOUNT=1
0055 SCAN(TOP)=ROOT
0056 40 IF(TOP.EQ.0) GO TO 70
0057 I=SCAN(TOP)
0058 IF(I.GT.0) GO TO 50
0059 I=-I
0060 NUMBER(I)=COUNT-LIST(I)
0061 TOP=TOP-1
0062 GO TO 40
0063 50 LIST(I)=COUNT
0064 COUNT=COUNT+1
0065 THREAD(PREV)=I
0066 PREV=I
0067 SCAN(TOP)=-I
0068 IF(KOUNT.GE.N)GO TO 40
0069 IX=POINT(I)
0070 IY=POINT(I+1)-1
0071 IF(IX.GT.IY) GO TO 40
0072 DO 60 J=IX,IY
0073 K=HEAD(J)
0074 IF(PRED(K).NE.0) GO TO 60
0075 PRED(K)=I
0076 LINK(K)=J
0077 TOP=TOP+1
0078 SCAN(TOP)=K
0079 KOUNT=KOUNT+1
0080 60 CONTINUE
0081 GO TO 40
0082 70 THREAD(PREV)=SINK
0083 THREAD(SINK)=SOURCE
0084 NUMBER(SINK)=1
0085 NUMBER(SOURCE)=N
0086 80 CONTINUE
0087 PI(I)=0
0088 PID(I)=0
0089 PID(0)=0

```

```

00910 PI(SINK)=LARGE
00920 PID(SINK)=0
00930 FLOW(I)=0
00940 FLOW(SINK)=V
00950 C=V*LARGE
00960 D=0.0
00970 I=SINK
90 K=THREAD(I)
00980 PI(K)=PI(PRED(K))+COST(LINK(K))
00990 PID(K)=PID(PRED(K))+BUD(LINK(K))
01000 FLOW(K)=0
01010 IF(K.EQ.PREV)GO TO 100
01020 I=K
01030 GO TO 90
01040 100 CONTINUE
01050 CODE=5
110 RSIGN=-1
01060 CMINIM=C
01070 DMINIM=D
01080 SIGN=-1
120 *****
* THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON-BASIC ARC PQ *
* FROM A SET OF NON-BASIC ARCS FOR FLOW AUGMENTATION. *
*****
01100 MUENAX=LARGE
01110 IF(D2.LT.DMINIM)SIGN=1
130 FLOWPO=0
01120 IF(DMINIM.GE.D1.AND.DMINIM.LE.D2.AND.C.GE.C1.AND.CODE.NE.5)GO TO
01130 1
01140 IF((ABS(D-D1).LE.S.OR.ABS(D2-D).LE.S).AND.CODE.LE.2)GO TO 460
01150 IF(ABS(C-C1).LE.S.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 880
01160 IF(ABS(D-D1).LE.S.OR.ABS(D-D2).LE.S).AND.(C.GE.C1)
01170 1
01180 .AND.(CODE.LE.2))GO TO 880
01190 IF(D.GE.D1.AND.D.LE.D2.AND.C.GE.C1.AND.C.LE.C2)GO TO 880
01200 ENTER=0
01210 ITER=ITER+1
01220 DO 230 I=1,N
01230 START=POINT(I)
01240 FINISH=POINT(I+1)-1
01250 IF(START.GT.FINISH)GO TO 230
01260 DO 230 J=START,FINISH
01270 K=HEAD(J)
01280 IF(CAP(J).LT.0)GO TO 140
01290 CBAR=COST(J)+PI(I)-PI(K)
01300 DBAR=BUD(J)+PID(I)-PID(K)
01310 STATUS=1
01320 GO TO 150
140 CBAR=PI(K)-PI(I)-COST(J)
01330 DBAR=PID(K)-PID(I)-BUD(J)
01340 STATUS=-1
150 GO TO(160,210,170,210,200)CODE
160 IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
01350 GO TO 180
170 IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
180 ENTER=J
01360 CMIN=CBAR
01370 DMIN=DBAR
01380 STATPO=STATUS
01390 IHEAD=K
01400 ITAIL=I
01410 PQ=J
190 GO TO 270
200 IF(CBAR.GE.0.OR.MUENAX.LE.CBAR)GO TO 230
01420 MUENAX=CBAR
01430 GO TO 220
210 IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
01440 MUE=(RSIGN*CBAR)/(SIGN*DBAR)
01450 IF(MUE.GE.MUENAX)GO TO 230
01460 MUENAX=MUE
220 ENTER=J
01470 CMIN=CBAR
01480 DMIN=DBAR
01490 STATPO=STATUS
01500 PQ=J
01510 IHEAD=K
01520 ITAIL=I
230 CONTINUE
01530 IF(CODE.EQ.5.AND.ENTER.NE.0)GO TO 270
01540 IF(CODE.EQ.5.AND.ENTER.EQ.0)CODE=1
01550 IF(ENTER.NE.0)GO TO 260
01560 IF(CODE.EQ.4.AND.ENTER.EQ.0)GO TO 880
01570 CODE=CODE+1
01580 IF(C1.LE.C.AND.C.LE.C2.AND.CODE.EQ.3)GO TO 880
01590 IF(CODE.NE.4)GO TO 120
240 SIGN=-SIGN
01600 RSIGN=-RSIGN
250 MUENAX=0
01610 GO TO 130
260 MAX1=(C2-C)/CMIN

```

```

1790 IF(CODE.GT.3)MAX1=(C1-C)/DMIN
1800 MAX2=(D1-D)/DMIN
1810 IF(D2.LT.DMIN)MAX2=(D2-D)/DMIN
1820 MAX=MAX1
1830 IF((MAX1.GT.MAX2.AND.MAX2.GT.0).OR.MAX1.LT.0)MAX=MAX2
1840 *****
1850 * THIS PART OF THE PROGRAM DETERMINS THE LEAVING ARC FROM THE CYCLE *
1860 * AND THE MINIMUM FLOW TO BE AUGMENTED IN THE CYCLE. *
1870 *****
1880 270 ICAP=CAP(ENTER)
1890 LEG=0
1900 FLOWPQ=IABS(CAP(PQ))
1910 IF(STATPQ.EQ.1)FLOWPQ=0
1920 IF(FLAG.EQ.0)GO TO 280
1930 FLOWPQ=IFLOW
1940 FLAG=0
1950 280 IF(ICAP.LT.0)GO TO 290
1960 STATUS=1
1970 IX=ITAIL
1980 IY=IHEAD
1990 DELTA1=ICAP-FLOWPQ
2000 GO TO 300
2010 290 ICAP=-ICAP
2020 CMIN=-CMIN
2030 DMIN=-DMIN
2040 STATUS=-1
2050 IX=IHEAD
2060 IY=ITAIL
2070 DELTA1=FLOWPQ
2080 DELTA=DELTA1
2090 RFLOW=FLOWPQ
2100 310 IF(IX.EQ.IY)GO TO 390
2110 IF(NUMBER(IX).GT.NUMBER(IY))GO TO 350
2120 J=LINK(IX)
2130 IF(J.LT.0)GO TO 320
2140 CHANGE=CAP(J)-FLOW(IX)
2150 GO TO 330
2160 320 CHANGE=FLOW(IX)
2170 330 IF(CHANGE.GE.DELTA)GO TO 340
2180 DELTA=CHANGE
2190 K=IX
2200 LEG=1
2210 340 IX=PRED(IX)
2220 GO TO 310
2230 350 J=LINK(IY)
2240 IF(J.LT.0)GO TO 360
2250 CHANGE=FLOW(IY)
2260 GO TO 370
2270 360 CHANGE=CAP(-J)-FLOW(IY)
2280 370 IF(CHANGE.GE.DELTA)GO TO 380
2290 DELTA=CHANGE
2300 K=IY
2310 LEG=-1
2320 380 IY=PRED(IY)
2330 GO TO 310
2340 390 CONTINUE
2350 IW=IX
2360 DELTA1=DELTA
2370 IF(CODE.NE.5)GO TO 410
2380 STATUS=STATPQ
2390 ENTER=PQ
2400 C=C+CMIN*STATUS*DELTA
2410 D=D+DMIN*STATUS*DELTA
2420 DIR=-1
2430 IF(D1.GT.D)DIR=1
2440 IF(DELTA1.LT.ICAP)GO TO 400
2450 CALL AUGMEN
2460 GO TO 110
2470 400 CALL UPDATE
2480 GO TO 110
2490 410 IF(DELTA1.LE.MAX)GO TO 450
2500 DELTA=MAX
2510 ADD=CMIN*STATUS*DELTA
2520 ADD1=DMIN*STATUS*DELTA
2530 C=C+ADD
2540 D=D+ADD1
2550 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
2560 ,C2,D2)
2570 C=C-ADD
2580 D=D-ADD1
2590 IF(OVER.EQ.1)GO TO 880
2600 C=C+ADD
2610 D=D+ADD1
2620 IF(ABS(D-D1).LE.S.OR.ABS(D-D2).LE.S).AND.CODE.GE.3)GO TO 420
2630 IF(ABS(D-D1).GE.S.AND.ABS(D-D2).GE.S)GO TO 420
2640 CALL AUGMEN
2650 CAP(ENTER)=-CAP(ENTER)
2660 GO TO 130
2670 420 DELTA=MAX1

```



```

12680 IF (MAX.EQ.MAX1) DELTA=MAX2
12690 IF (DELTA.LT.0.OR.DELTA.GE.DELTA1) GO TO 440
12700 DELTA=DELTA-MAX
12710 ADD1=C*MIN*STATUS*DELTA
12720 ADD2=D*MIN*STATUS*DELTA
12730 C=C+ADD1
12740 D=D+ADD2
12750 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
12760 ,C2,D2)
12770 IF (OVER.EQ.1) GO TO 430
12780 DELTA=DELTA+MAX
12790 CALL AUGMEN
12800 GO TO 880
12810 430 DELTA=MAX
12820 C=C-ADD1
12830 D=D-ADD2
12840 CALL AUGMEN
12850 GO TO 880
12860 440 DELTA=DELTA1-MAX
12870 450 ADD=C*MIN*STATUS*DELTA
12880 ADD1=D*MIN*STATUS*DELTA
12890 C=C+ADD
12900 D=D+ADD1
12910 DELTA=DELTA1
12920 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
12930 ,C2,D2)
12940 C=C-ADD
12950 D=D-ADD1
12960 IF (OVER.EQ.1.AND.(DELTA.EQ.(DELTA1-MAX))) GO TO 430
12970 IF (OVER.EQ.1) GO TO 880
12980 C=C+ADD
12990 D=D+ADD1
13000 IF (LEG.EQ.0) CALL AUGMEN
13010 IF (LEG.NE.0) CALL UPDATE
13020 IF (CODE.EQ.4) GO TO 250
13030 GO TO 120
13040 460 CBPO=C*MIN
13050 DBPO=D*MIN
13060 TAILPO=ITAIL
13070 HEADPO=IHEAD
13080 FLOWPO=DELTA
13090 IF (STATPO.EQ.-1) FLOWPO=-CAP(PO)-DELTA
13100 *****
13110 * THIS PART OF THE PROGRAM SELECTS AN ARC RS FROM A SET OF *
13120 * NON-BASIC ARCS FOR FLOW AUGMENTATION IN TWO CYCLES SIMULTANEOUSLY *
13130 *****
13140 470 CODE2=1
13150 480 RS=0
13160 DO 490 I=1,N
13170 490 SUM(I)=0.0
13180 IF (C.GE.C1.AND.C.LE.C2) GO TO 880
13190 ITER=ITER+1
13200 DO 520 I=1,N
13210 START=POINT(I)
13220 FINISH=POINT(I+1)-1
13230 IF (START.GT.FINISH) GO TO 520
13240 DO 520 J=START,FINISH
13250 K=HEAD(J)
13260 IF (CAP(J).LT.0) GO TO 500
13270 CBAR=COST(J)+PI(I)-PI(K)
13280 DBAR=BUD(J)+PID(I)-PID(K)
13290 STATUS=1
13300 GO TO 510
13310 500 CBAR=PI(K)-PI(I)-COST(J)
13320 DBAR=PID(K)-PID(I)-BUD(J)
13330 STATUS=-1
13340 510 IF (DBAR.EQ.0) GO TO 520
13350 IF (DBAR/DBPO.GE.0) GO TO 520
13360 IF ((CBPO-(DBPO*CBAR)/DBAR).LE.0) GO TO 520
13370 RS=J
13380 TAILRS=I
13390 HEADRS=K
13400 STATRS=STATUS
13410 RATIO=-DBPO/DBAR
13420 CBRS=CBAR
13430 DBRS=DBAR
13440 FLOWRS=0.0
13450 IF (STATRS.EQ.-1) FLOWRS=-CAP(RS)
13460 GO TO 530
13470 520 CONTINUE
13480 IF (CODE2.EQ.2.AND.RS.EQ.0) GO TO 530
13490 CBPO=-CBPO
13500 DBPO=-DBPO
13510 CAP(PO)=-CAP(PO)
13520 STATPO=-STATPO
13530 CODE2=CODE2+1
13540 GO TO 480
13550 530 CODE=4
13560 CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG

```

```

03580      IF(CBPQ.GT.0)GO TO 540
03590      CBPQ=-CBPQ
03600      DBPQ=-DBPQ
03610      CAP(PQ)=-CAP(PQ)
03620      IF(DBPQ.LT.0.AND.DIR.EQ.-1)RFLAG=-1
03630      IF(DBPQ.GT.0.AND.DIR.EQ.1)RFLAG=+1
03640      IF(RFLAG.NE.0)SIGN=-SIGN
03650      FLAG=1
03660      IFLOW=FLOWPQ
03670      GO TO 240
03680
03690 *****
03700 * THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE *
03710 * MINIMUM FLOW TO BE AUGMENTED IN THE TWO CYCLES SIMULTANEOUSLY*
03720 *****
03730
03740 550      CODE1=1
03750      DELTA1=FLOWPQ
03760      IF(STATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
03770      LEG=0
03780      DELTA2=IABS(CAP(RS))/RATIO
03790      IF(DELTA2.LT.DELTA1)DELTA1=DELTA2
03800 560      ENTER=PQ
03810      STATUS=STATPQ
03820      RIN=1
03830      INC=1
03840      ITAIL=TAILPQ
03850      IHEAD=HEADPQ
03860      IF(STATUS.EQ.-1)GO TO 580
03870      IX=ITAIL
03880      IY=IHEAD
03890      GO TO 590
03900 580      IX=IHEAD
03910      IY=ITAIL
03920      IF(IX.EQ.IY)GO TO 750
03930      IF(NUMBER(IX).GT.NUMBER(IY))GO TO 670
03940      IF(CODE1.EQ.3)GO TO 620
03950      J=LINK(IX)
03960      IF(J.LT.0)GO TO 600
03970      SUM(IX)=SUM(IX)+RIN
03980      GO TO 610
03990      SUM(IX)=SUM(IX)-RIN
04000 610      IF(CODE1.EQ.1)GO TO 660
04010 620      IF(SUM(IX))630,660,640
04020 630      DELTA2=-FLOW(IX)/SUM(IX)
04030      GO TO 650
04040 640      J=IABS(LINK(IX))
04050      DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
04060 650      IF(DELTA2.GE.DELTA1)GO TO 660
04070      DELTA1=DELTA2
04080      LEG=INC
04090      K1=IX
04100      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
04110 660      IX=PRED(IX)
04120      GO TO 590
04130 670      IF(CODE1.GT.2)GO TO 700
04140      J=LINK(IY)
04150      IF(J.LT.0)GO TO 680
04160      SUM(IY)=SUM(IY)-RIN
04170      GO TO 690
04180      SUM(IY)=SUM(IY)+RIN
04190 680      IF(CODE1.EQ.1)GO TO 740
04200 700      IF(SUM(IY))710,740,720
04210 710      DELTA2=-FLOW(IY)/SUM(IY)
04220      GO TO 730
04230 720      J=IABS(LINK(IY))
04240      DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
04250 730      IF(DELTA2.GE.DELTA1)GO TO 740
04260      DELTA1=DELTA2
04270      LEG=-INC
04280      K1=IY
04290      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
04300 740      IY=PRED(IY)
04310      GO TO 590
04320 750      IF(CODE1.EQ.1)I1=IX
04330      IF(CODE1.EQ.2)I2=IX
04340      CODE1=CODE1+1
04350      IF(CODE1.GT.3)GO TO 760
04360      IF(CODE1.EQ.3)GO TO 560
04370      ENTER=RS
04380      ITAIL=TAILRS
04390      IHEAD=HEADRS
04400      STATUS=STATRS
04410      RIN=RATIO
04420      INC=2
04430      GO TO 570
04440 760      DELTA2=(C1-C)/((CBPQ+RATIO*CBRS)
04450      IF(DELTA2.GT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
04460      C=C+DELTA1*CBPQ+RATIO*DELTA1*CBRS
04470      D=D+DELTA1*DBPQ+RATIO*DELTA1*DBRS

```

```

*****
* THIS PART OF THE PROGRAM CALLS AN APPROPRIATE UPDATE SUBROUTINE
* FOR UPDATING FLOW, INDICES AND DUAL VARIABLES AND FINDS THE ARC PQ *
*****
IF(IABS(LEG).GT.0)GO TO 800
DELTA=DELTA1*RATIO
ITAIL=TAILRS
IHEAD=HEADRS
STATUS=STATRS
FLOWRS=FLOWRS+STATRS*DELTA
IW=IW2
ENTER=RS
CALL AUGMEN
DELTA=DELTA1
ITAIL=TAILPQ
IHEAD=HEADPQ
STATUS=STATPQ
FLOWPQ=FLOWPQ+STATPQ*DELTA
IW=IW1
ENTER=PQ
CALL AUGMEN
IF(FLOWRS.NE.0.AND.FLOWRS.NE.IABS(CAP(RS)))GO TO 780
STATPQ=-STATPQ
CBPQ=-CBPQ
DBPQ=-DBPQ
770 IF(FLOWPQ.LE.S.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 470
STATPQ=-STATPQ
CAP(PQ)=-CAP(PQ)
CBPQ=-CBPQ
DBPQ=-DBPQ
GO TO 470
780 FLOWPQ=FLOWRS
CBPQ=-CBRS
DBPQ=-DBRS
PQ=RS
TAILPQ=TAILRS
HEADPQ=HEADRS
STATPQ=-STATRS
790 IF(FLOWPQ.LE.S.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 470
CAP(PQ)=-CAP(PQ)
STATPQ=-STATPQ
CBPQ=-CBPQ
DBPQ=-DBPQ
GO TO 470
800 IF(IABS(LEG).EQ.2)GO TO 840
ITAIL=TAILRS
IHEAD=HEADRS
STATUS=STATRS
DELTA=DELTA1*RATIO
FLOWRS=FLOWRS+STATRS*DELTA
IW=IW2
ENTER=RS
CALL AUGMEN
ITAIL=TAILPQ
IHEAD=HEADPQ
STATUS=STATPQ
DELTA=DELTA1
IW=IW1
ENTER=PQ
CMIN=CBPQ
DMIN=DBPQ
IF(STATUS.EQ.1)GO TO 810
CMIN=-CMIN
DMIN=-DMIN
810 ICAP=IABS(CAP(PQ))
K=K1
RFLW=FLOWPQ
CALL UPDATE
CAP(LEAVE)=-IABS(CAP(LEAVE))
IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
FLOWPQ=FLOWRS
PQ=RS
TAILPQ=TAILRS
HEADPQ=HEADRS
CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
IF(STATRS.EQ.1)GO TO 820
CBPQ=-CBPQ
DBPQ=-DBPQ
820 STATPQ=-STATRS
CBPQ=-CBPQ
DBPQ=-DBPQ
830 GO TO 790
840 ENTER=PQ
ITAIL=TAILPQ
IHEAD=HEADPQ

```



```

DELTA=DELTA1
IW=IW1
FLOWPQ=FLOWPQ+STATPQ*DELTA1
CALL AUGMEN
ITAIL=TAILRS
IHEAD=HEADRS
STATJS=STATRS
DELTA=DELTA1*RATIO
IN=IN2
LEG=LEG/2
ENTER=RS
K=K1
CMIN=CBRS
DMIN=DBRS
IF(STATUS.EQ.1)GO TO 850
CMIN=-CMIN
DMIN=-DMIN
ICAP=IABS(CAP(RS))
REFLOW=FLOWRS
CALL UPDATE
CAP(LEAVE)=-IABS(CAP(LEAVE))
IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
IF(STATPQ.EQ.1)GO TO 860
CBPQ=-CBPQ
DBPQ=-DBPQ
STATPQ=-STATPQ
CBPQ=-CBPQ
DBPQ=-DBPQ
GO TO 770
870 CALL RTIME(TIME2)
880 TIME=TIME2-TIME1
WRITE(22,890),C,D,TIME,ITER
890 FORMAT(1X,'COST=',F14.4,'BUDGET=',F14.4,'TIME=',I10,'ITER=',I8)
STOP
END

```

\*\*\*\*\*  
 \* THIS SUBROUTINE UPDATES THE OBJECTIVE FUNCTION OF INTERVAL \*  
 \* GOAL PROGRAMMING \*  
 \*\*\*\*\*

```

SUBROUTINE UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,
1  RFLAG,C2,D2)
REAL MIN,MINIM
INTEGER OVER
IF(DELTA.EQ.0)GO TO 50
IF(C.GT.C2.OR.C.LT.C1.OR.D.LT.D1.OR.D.GT.D2)GO TO 10
OVER=1
GO TO 50
10 IF(C.GT.C1.AND.C.LT.C2)GO TO 50
IF(D.GT.D1.AND.D.LT.D2)GO TO 50
IF(RFLAG)70,20,30
IF(DIR)30,50,70
20 IF(C.GE.C2)GO TO 60
30 IF(C.GE.C2)GO TO 60
MIN=W1*(C1-C)+W4*(D-D2)
40 OVER=1
IF(MIN.GE.MINIM)GO TO 50
MINIM=MIN
OVER=0
C TYPE*,C,D,C1,D1,C2,D2,MIN,MINIM,OVER
50 RETURN
60 MIN=W3*(C-C2)+W4*(D-D2)
GO TO 40
70 IF(C.GE.C2)GO TO 80
MIN=W1*(C1-C)+W2*(D1-D)
GO TO 40
80 MIN=W3*(C-C2)+W2*(D1-D)
GO TO 40
END
  
```

```

*****
* THIS SUBROUTINE UPDATES THE DUAL VARIABLES PI'S AND PID'S, *
* THREAD INDICES, NO. OF SUCCESSORS, PREDECESSOR INDICES AND *
* FLOW *
*****

```

```

SUBROUTINE UPDATE
INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
1 FINISH,ENTER,V,STATUS,O,OP,Z,X,R,
2 TIME1,TIME2,TIME,CMIN,TR,
3 PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
4 SCAN(405),HEAD(3000),CAP(3000),CUST(3000),BUD(0:3000),
5 PO,RS,TAILPO,TAILRS,HEADPO,HEADRS,CODE,CODE1,CODE2
REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,JFLOW
DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
1 LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
2 ,LEAVE,RFLOW
IFLOW=RFLOW+STATUS*DELTA
C IF(STATUS.EQ.-1)IFLOW=ICAP-DELTA
ILINK=-STATUS*LEG*ENTER
IF(ILINK.GT.0)GO TO 610
O=ITAIL
OP=IHEAD
CMIN=-CMIN
DMIN=-DMIN
GO TO 620
610 O=IHEAD
620 OP=ITAIL
CHANGE=LEG*DELTA
KP=PRED(K)
LEAVE=IABS(LINK(K))
KODE=LEG
IF(LINK(K).LT.0)KODE=-KODE
IF(KODE.EQ.1)CAP(LEAVE)=-CAP(LEAVE)
CAP(ENTER)=ICAP
IPRED=OP
INUMB=NUMBER(K)
L=0
I=0
625 JFLOW=FLOW(I)
JLINK=LINK(I)
JPRED=PRED(I)
JSUCC=NUMBER(I)-L
Z=I
X=THREAD(I)
COUNT=2
630 IF(COUNT.GT.JSUCC)GO TO 640
PI(X)=PI(X)+CMIN
PID(X)=PID(X)+DMIN
Z=X
X=THREAD(X)
COUNT=COUNT+1
GO TO 630
640 R=JPRED
650 TR=THREAD(R)
IF(TR.EQ.I)GO TO 660
R=TR
GO TO 650
660 FLOW(I)=IFLOW
LINK(I)=ILINK
NUMBER(I)=INUMB
PRED(I)=IPRED
PI(I)=PI(I)+CMIN
PID(I)=PID(I)+DMIN
THREAD(R)=X
IF(I.EQ.K)GO TO 690
THREAD(Z)=JPRED
IF(JLINK.LT.0)GO TO 670
IFLOW=JFLOW+CHANGE
GO TO 680
670 IFLOW=JFLOW-CHANGE
680 ILINK=-JLINK
INUMB=INUMB-JSUCC
L=L+JSUCC
IPRED=I
I=JPRED
GO TO 625
690 THREAD(Z)=THREAD(OP)
THREAD(OP)=O
CHANGE=-CHANGE
NUMBQ=NUMBER(O)
I=OP
710 IF(I.EQ.IN)GO TO 740
MM=MM+1
IF(LINK(I).LT.0)GO TO 720
FLOW(I)=FLOW(I)+CHANGE
NUMBER(I)=NUMBER(I)+NUMBQ
I=PRED(I)

```

0900	720	FLOW(I)=FLOW(I)-CHANGE
0910		NUMBER(I)=NUMBER(I)+NUMBQ
0920		I=PRED(I)
0930		GO TO 710
0940	740	I=KP
0950		CHANGE=-CHANGE
0960	750	IF(I.EQ.IW)RETURN
0970		IF(LINK(I).LT.0)GO TO 760
0980		FLOW(I)=FLOW(I)+CHANGE
0990		NUMBER(I)=NUMBER(I)-NUMBQ
1000		I=PRED(I)
1010		GO TO 750
1020	760	FLOW(I)=FLOW(I)-CHANGE
1030		NUMBER(I)=NUMBER(I)-NUMBQ
1040		I=PRED(I)
1050		GO TO 750
1060		END

```

0001 *****
0002 * THIS SUBROUTINE UPDATES THE FLOW *
0003 *****
0004 SUBROUTINE AUGMEN
0005 INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
0006 1 FINISH,ENTER,V,STATUS,Q,Z,X,R,
0007 2 TIME1,TIME2,TIME,CMIN,TR,
0008 3 PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
0009 4 SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
0010 5 P,RS,TAILPO,TAILRS,HEADPO,HEADRS,CODE,CODE1,CODE2
0011 REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW1,JFLOW
0012 DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
0013 COMMON IAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
0014 1 LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
0015 2 ,LEAVE,RFLOW
0016 CAP(ENTER)=-CAP(ENTER)
0017 CHANGE=DELTA*STATUS
0018 I=IAIL
0019 410 IF(I.EQ.IW)GO TO 440
0020 J=LINK(I)
0021 IF(J.LT.0)GO TO 420
0022 FLOW(I)=FLOW(I)+CHANGE
0023 I=PRED(I)
0024 GO TO 410
0025 420 FLOW(I)=FLOW(I)-CHANGE
0026 I=PRED(I)
0027 GO TO 410
0028 440 I=IHEAD
0029 450 IF(I.EQ.IW)RETURN
0030 J=LINK(I)
0031 IF(J.LT.0)GO TO 460
0032 FLOW(I)=FLOW(I)-CHANGE
0033 I=PRED(I)
0034 GO TO 450
0035 460 FLOW(I)=FLOW(I)+CHANGE
0036 I=PRED(I)
0037 GO TO 450
0038 END

```

```

*****
* THIS PART OF THE PROGRAM ( SUBROUTINES NG4 , STAR AND
* FUNCTION IRAN ) GENERATES THE NETWORK.
*****

```

```

SUBROUTINE NG4(WIDTH,LENGTH,N,M,S,T,POINT,HEAD)

```

```

*****
* THIS SUBROUTINE GENERATES THE HEAD NODES OF ARCS
*****

```

```

INTEGER S,T,COUNT,WIDTH,
1 ALIST(3000),BLIST(3000),POINT(405),RPOINT(405),
2 TAIL(3000),HEAD(3000),CAP(3000),TRACE(3000)

```

```

LARGE=1000000
N=LENGTH*WIDTH+2

```

```

S=1
T=N
IA=LARGE
IB=LARGE
IC=0
ID=100
IW=2

```

```

IX=WIDTH+1
IY=N-WIDTH
IZ=N-1

```

```

DO 10 I=1,N

```

```

ALIST(I)=0

```

```

CONTINUE

```

```

COUNT=WIDTH*(LENGTH+1)+1

```

```

DO 20 J=COUNT,M

```

```

ITAIL=IRAN(1,N-1)

```

```

ALIST(ITAIL)=ALIST(ITAIL)+1

```

```

CONTINUE

```

```

COUNT=0

```

```

DO 30 I=IW,IX-1

```

```

COUNT=COUNT+1

```

```

TAIL(COUNT)=1

```

```

HEAD(COUNT)=1

```

```

CAP(COUNT)=IRAN(IA,IB)

```

```

CONTINUE

```

```

DO 150 K=1,IZ

```

```

DO 50 I=1,N

```

```

BLIST(I)=0

```

```

I1=MINO(K+WIDTH,N)

```

```

BLIST(I1)=1

```

```

IF(ALIST(K).EQ.0)GO TO 110

```

```

DO 100 J=1,ALIST(K)

```

```

IHEAD=IRAN(2,N)

```

```

IF(BLIST(IHEAD).EQ.1)GO TO 90

```

```

IF(IHEAD.EQ.K)GO TO 90

```

```

BLIST(IHEAD)=1

```

```

CONTINUE

```

```

DO 120 I=N,2,-1

```

```

IF(BLIST(I).EQ.0)GO TO 120

```

```

COUNT=COUNT+1

```

```

TAIL(COUNT)=K

```

```

HEAD(COUNT)=1

```

```

IF(TAIL(COUNT).EQ.1.AND.HEAD(COUNT).EQ.IX)GO TO 130

```

```

IF(HEAD(COUNT).EQ.N.AND.HEAD(COUNT)-TAIL(COUNT).LE.WIDTH)

```

```

1 GO TO 130

```

```

CAP(COUNT)=IRAN(IC,ID)

```

```

GO TO 120

```

```

CAP(COUNT)=IRAN(IA,IB)

```

```

CONTINUE

```

```

CONTINUE

```

```

CALL STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)

```

```

RETURN
END

```

10

20

30

50

90

100

110

130

120

150

```

00030 *****
00040 * THIS SUBROUTINE GENERATES POINTERS OF THE NETWORK *
00050 *****
00060 SUBROUTINE STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)
00070 INTEGER COUNT,POINT(405),RPOINT(405),ALIST(3000),BLIST(3000),
00080 1 TAIL(3000),HEAD(3000),TRACE(3000),CAP(3000)
00090
00100 DO 10 I=1,N
00110 ALIST(I)=0
00120 CONTINUE
00130 10 POINT(I)=1
00140 K=1
00150 DO 20 J=1,M
00160 ALIST(HEAD(J))=ALIST(HEAD(J))+1
00170 IF(TAIL(J).EQ.K)GO TO 20
00180 K=K+1
00190 POINT(K)=J
00200 CONTINUE
00210 20 POINT(N)=M+1
00220 POINT(N+1)=M+1
00230 C CALL SORT(N,M,POINT,TAIL,HEAD,CAP)
00240 BLIST(1)=1
00250 RPOINT(1)=1
00260 DO 30 I=2,N
00270 BLIST(I)=BLIST(I-1)+ALIST(I-1)
00280 RPOINT(I)=BLIST(I)
00290 CONTINUE
00300 30 RPOINT(N+1)=M+1
00310 DO 40 J=M,1,-1
00320 JHEAD=HEAD(J)
00330 TRACE(BLIST(JHEAD))=J
00340 BLIST(JHEAD)=BLIST(JHEAD)+1
00350 CONTINUE
00360 40 RETURN
00370 END
00380

```

\*\*\*\*\*  
\* THIS FUNCTION GENERATES A RANDOM NUMBER BETWEEN IC AND ID. \*  
\*\*\*\*\*

6.18  
FUNCTION IRAN(IC, ID)  
IRAN=IC+(ID-IC+1)\*RAN(X)  
IF(IRAN.GT.ID)IRAN=ID  
RETURN  
END



A 87470